SÍNTESIS AUTOMÁTICA DE ESTRATEGIAS DE CONTROL A TRAVÉS DE LÓGICAS TEMPORALES Y REDES DE PETRI PARA LA PLANIFICACIÓN DE MOVIMIENTOS EN SISTEMAS AUTÓNOMOS

JORGE LUIS MARTÍNEZ VALENCIA

Proyecto de grado presentado como requisito parcial
para aspirar al título de Magíster en Ingeniería Eléctrica

Director
Ing Mauricio Holguín L, M.Sc, Ph.D(C)

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
MAESTRÍA EN INGENIERÍA ELÉCTRICA
PEREIRA
2018

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del Presidente del jurado

_____

Firma del jurado 1 - Evaluador

_____

Firma del jurado 2 - Evaluador

_____

Firma del jurado 3 - Director

Pereira, 8 de agosto de 2018

Dedicado a mi esposa, por siempre creer en mi y alentarme a superar mis propias expectativas.

Agradecimientos...

A mi tutor, el Ing. Mauricio Holguín Londoño, por su paciencia, dedicación y compromiso para con mi formación académica, profesional y personal.

Al ingeniero Germán Andrés Holguín, por su asesoría en temas fundamentales de la investigación y por avivar mi pundonor con su altruista ayuda y disposición.

Se agradece a la Universidad Tecnológica de Pereira por su apoyo a través de la convocatoria interna para financiación de grupos de investigación de la Vicerrectoría de Investigaciones, Innovación y Extensión, así como a la Maestría en Ingeniería Eléctrica y al Grupo de Investigación en Gestión de Sistemas Eléctricos, Electrónicos y Automáticos.

Este trabajo de grado es un producto esperado del proyecto "METODOLOGÍA PARA SINTETIZAR CONTINUAMENTE AUTÓMATAS DE ESTADOS FINITOS A TRAVÉS DEL USO DE LENGUAJES FORMALES Y LÓGICAS TEMPORALES, APLICADA A LA NAVEGACIÓN AUTÓNOMA TERRESTRE EN AMBIENTES PARCIALMENTE CONTROLADOS" adscrito a la Vicerrectoría de Investigaciones, Innovación y Extensión bajo el código de proyecto 6-17-2.

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE FIGURAS

Parte I

INTRODUCCIÓN

En este trabajo se presenta una metodología para la síntesis automática de estrategias de control a través de lógicas temporales y redes de Petri para la planificación de movimientos en sistemas autónomos. Se inicia recolectando los trabajos más representativos en el estado del arte, posteriormente se hace un análisis para seleccionar la Lógica Temporal Lineal (LTL) como lenguaje para especificación de tareas robóticas. En la tercera fase se plantea un procedimiento que permite, a partir de una fórmula LTL, calcular rutas óptimas en términos de transiciones para un equipo de múltiples agentes modelados como una red de Petri. Finalmente se presenta la metodología que recoge 4 algoritmos evaluados en tres casos propuestos (caso1: un robot con obstáculos fijos, caso 2: un robot con obstáculos móviles, caso 3: múltiples robots y obstáculos fijos). La metodología propuesta sevalida con una serie de 20 simulaciones sobre las cuales se analizan los datos obtenidos en cada ejecución.

1. DEFINICIÓN DEL PROBLEMA

La complejidad presente en la tarea de hacer que las máquinas puedan establecer comunicación y comprender el entorno humano ha retenido el desarrollo de tecnologías más complejas que puedan conducir tareas autónomas con desenvolvimiento en entornos físicos compartidos con seres humanos. Por ejemplo, la competencia europea SAUC-E (Student Autonomous Underwater Vehicle Challenge-Europe) reta a académicos a desarrollar vehículos submarinos autónomos (AUV) [1]. Dentro de las aplicaciones más comunes se encuentra la elaboración de cartografía del fondo del mar, la medición de propiedades del agua con fines ambientales, la detección y eliminación de minas submarinas, el reconocimiento del campo de batalla y la protección de puertos marítimos. Por otro lado, estos ambientes dinámicos, plantean desafíos únicos para el desarrollo de técnicas de control y verificación, ya que existen factores a tener en cuenta como: la dinámica del ruido del sistema, el acondicionamiento de sensores y trans-

ductores, las estrictas condiciones de seguridad que se deben asumir al interactuar con seres humanos, la interacción con usuarios inexpertos y otros.

Los entornos físicos descritos, compartidos con humanos para el desenvolvimiento de un sistema autónomo, llevan consigo un conjunto de subcategorías de entornos parametrizados, conocidos y aceptados. Por ejemplo, en un entorno terrestre se puede definir una categoría "Urbana"[2], con parámetros que determinan la manera correcta en la que se debe desenvolver un sistema autónomo en medio de un ambiente citadino en el que se debe ejecutar tareas como la evasión de obstáculos en la vía, estacionar y el conjunto de normas de tráfico vehicular que apliquen a la zona urbana específica y además interpretar señales como el paso por intersecciones, las paradas de emergencia, entre otras. Si bien la conducción autónoma ha demostrado ser confiable y segura, la convivencia de dicha tecnología con los autos convencionales está presentando diversos conflictos [3], debido a la gran complejidad y dificultad para interpretar el entorno humano. Los autómatas a menudo no logran hacer lo que se espera que hagan; se van por el camino equivocado que conduce a los obstáculos y a veces simplemente permanecen estáticos en un lugar hasta que se logra identificar que es lo que está mal. En el proceso de diseño actual de los sistemas robóticos, especialmente en la toma de decisiones y los algoritmos de control, un ingeniero diseña el sistema, lo pone a prueba, lo ajusta en caso de que se encuentre un error, y luego se pone a prueba una vez más, hasta que él o ella decide que el sistema es lo suficientemente confiable. Si el comportamiento del robot necesita ser cambiado, este proceso es lento y propenso a errores. Los sistemas modernos de ingeniería planteados para afrontar este tipo de desafíos poseen un estrecho vínculo entre elementos computacionales y físicos, lo que hace que su diseño y verificación sean cada vez más complejos debido al entrelazamiento entre las lógicas de alto nivel encargadas de la planificación de los movimientos y las dinámicas de bajo nivel [4].

La planificación de movimientos en los sistemas autónomos se enfoca en calcular las trayectorias necesarias para que un sistema cumpla una tarea [5, 6]. Este problema es estudiado no solo para el caso de un sistema autónomo, sino también, para el caso de múltiples agentes [7, 8, 9]. Los robots pueden variar desde los brazos manipuladores utilizados en la producción,

hasta los empleados en cirugías y los vehículos autónomos utilizados en búsqueda y rescate o en exploración planetaria, y hasta sillas de ruedas inteligentes para personas con discapacidad. Estos sistemas robóticos están sujetos a restricciones mecánicas (sistemas holonómicos y no holonómicos) y tienen capacidades de computación, detección y comunicación limitadas, por ejemplo, un robot similar a un automóvil no se puede mover hacia los lados o un avión no se puede detener en cualquier lugar. Los entornos pueden estar llenos de obstáculos que posiblemente se muevan o cambien de forma (entorno no estructurado). Uno de los principales desafíos en esta área es el desarrollo de un marco computacionalmente eficiente que cumpla con las restricciones físicas del robot y con la complejidad del entorno, al tiempo que permita un amplio espectro de especificaciones de tareas [10]. En los problemas tradicionales de planificación de movimiento [11, 6], las especificaciones de tareas se dan simplemente como "pasar de A a B mientras se evitan obstáculos", donde A y B son las dos configuraciones o regiones de interés en el espacio de trabajo del robot. Sin embargo, una gran cantidad de aplicaciones robóticas requieren lenguajes de especificación más expresivos. Como alternativa para avanzar sobre estos problemas, existen las llamadas "Tecnologías del lenguaje", que se entienden como la aplicación de los conocimientos sobre la lengua al desarrollo de sistemas informáticos que puedan reconocer y generar lenguaje humano. Para tal fin, las tecnologías del lenguaje necesitan generar modelos formales y generales que capten la estructura del lenguaje natural y que sean eficaces desde el punto de vista computacional [12].

Los lenguajes formales, ámbito de investigación que surge a partir de la propuesta de Noam Chomsky acerca de las jerarquías de lenguajes [13], están sólidamente asentados en la teoría de sistemas formales y en la lingüística algebraica, entendida como disciplina que maneja herramientas de matemática formal para la descripción de fenómenos lingüísticamente reseñables [14]. La teoría de lenguajes formales y la teoría de autómatas, también llamada teoría algebraica de máquinas, constituyen los fundamentos de las ciencias del control de autómatas y por ende están inseparablemente unidas. Permiten además estudiar de modo sistemático las máquinas que realizan un procesamiento de la información y que actúan de manera discreta, esto es, la información se supone codificada a partir de un conjunto finito de símbolos que el

autómata trata secuencialmente, en la medida que los conjuntos de símbolos de entrada y de resultados de un sistema computacional cualesquiera pueden ser considerados como lenguajes [15].

Dentro de la teoría de autómatas sobresalen los autómatas de estados finitos deterministas, que son aquellos que sólo pueden estar en un único estado después de leer cualquier secuencia de entradas [16]. Para definir estos autómatas relacionándolos con un contexto específico, se utiliza un alfabeto proposicional con el cual se hacen proposiciones atómicas que poseen unos valores de verdad [17] y a través de la relación de estas proposiciones se genera el lenguaje del autómata; todo esto apoyado en la lógica clásica, que es aquella en la que una proposición es verdadera o falsa.

Sin embargo, al tratar de analizar bajo la lógica clásica las proposiciones cuyos valores de verdad dependen del momento (tiempo) y el espacio (sitio) en que se evalúan, es decir, cuando el controlador está cambiando dinámicamente debido a las diversas transiciones probables que se pueden dar [18] y por ende los valores de verdad de las proposiciones también deben cambiar respecto al tiempo o ubicación, se genera una explosión combinacional de estados [19] que hace más difícil, complejo y poco eficiente el desarrollo de un autómata debido a que no se puede considerar determinista. Tratando de dar solución a este problema, algunos autores aprovechan la disponibilidad de poderosas herramientas de comprobación de modelos para sintetizar protocolos de control complejos para la generación de autómatas [20, 21]. Estos enfoques normalmente generan estrategias de control en lazo abierto, es decir, no pueden satisfacer los comportamientos reactivos que dependen del estado actual del entorno en el que se desenvuelve el sistema o que manejan condiciones iniciales inciertas. Si además de las variaciones de los valores de verdad de las proposiciones atómicas con respecto al tiempo se suma que puedan existir, no uno sino varios sistemas autónomos disponibles, el problema de especificación y despliegue de tareas es aún más desafiante.

2. JUSTIFICACIÓN

Algunos trabajos como [22, 23], sugieren que las lógicas temporales, como la lógica temporal lineal (LTL) y la lógica de árbol de cálculo (CTL) [24], pueden ser utilizadas como lenguajes de especificación enriquecida en sistemas autónomos, como la robótica móvil. Dichas lógicas temporales son casos particulares sobre lógicas modales, que fueron desarrolladas inicialmente por filósofos para capturar valores de verdad proposicionales que cambian en mundos diferentes y, en particular, en momentos diferentes. La aplicación principal de las lógicas temporales es en ciencias de la computación, donde se usan habitualmente para especificar las propiedades de corrección de los programas de computadora y circuitos digitales. Las propiedades más utilizadas son la fórmula de seguridad en la que se establece que algo malo nunca ocurre, y la de existencia, que expresa que algo bueno, como atender una solicitud, eventualmente ocurre. La verificación de modelos es un área bien establecida en la cual el objetivo es desarrollar algoritmos computacionalmente eficientes para verificar tales propiedades frente a modelos matemáticos de programas de computadora y circuitos digitales. Aunque las herramientas existentes de comprobación de modelos como NuSMV [25] y SPIN [26] pueden manejar problemas muy grandes, se basan en una propiedad fundamental de los sistemas analizados: el número de estados es finito.

La Lógica Temporal Lineal (LTL) a menudo se encuentra como un formalismo para expresar tareas de alto nivel para sistemas autónomos [7, 27, 28]. Normalmente se comienza obteniendo un modelo de eventos discretos para el sistema autónomo, se combina este modelo con un autómata que corresponde a la fórmula LTL y luego se razona sobre tales abstracciones. Las tareas de LTL pueden referirse a un solo robot [7], pueden especificar requisitos individuales para robots móviles [27] o pueden imponer una especificación global para un equipo robótico [28]. En [29] se propone un algoritmo iterativo que planifica los movimientos de un equipo de robots que se desenvuelve en un espacio de trabajo modelado como una red de Petri con el fin de alcanzar un objetivo especificado como una fórmula LTL, la parte principal del algoritmo está representada por formulaciones de programación matemática específicas que producen

trayectorias para los robots, sin considerar las colisiones entre ellos.

Es por todo lo descrito previamente que se plantea como hipótesis de investigación el determinar si es posible desarrollar una metodología que permita sintetizar automáticamente estrategias de control a través del uso de lógicas temporales y redes de Petri para la planificación de movimientos de sistemas autónomos que se desenvuelven en un mismo espacio de trabajo, mitigando el problema de explosión combinacional de estados que se presenta al utilizar más de un robot para alcanzar la misma tarea.

La determinación de si es posible, o no, el desarrollo de esta metodología permitirá abonar el terreno que deben recorrer los diferentes investigadores a nivel nacional e internacional en cuanto a la utilización de lenguajes formales, lógicas temporales, redes de Petri y autómatas finitos para tratar el problema de explosión combinacional de estados asociado a la sintetización de políticas de control para procesos de trabajo colaborativo en sistemas autónomos que se presentan en diversas aplicaciones que hacen parte de la cotidianidad. Algunas aplicaciones que pueden ser analizadas como procesos colaborativos y que podrían mejorar la interacción del ser humano con su entorno son: diseño y control de sistemas de transporte masivo, control activo y dinámico de tráfico urbano, diseño de sistemas de control de ascensores, cobertura de áreas de interés con fines de seguridad, sistemas de manufactura flexible, entre otros.

3. OBJETIVOS

3.1. Objetivo General

Desarrollar una metodología que permita sintetizar automáticamente estrategias de control a través del uso de lógicas temporales y redes de Petri para la planificación de movimientos de sistemas autónomos que se desenvuelven en un mismo espacio de trabajo.

3.2. Objetivos Específicos

- Indagar en el estado del arte, la aplicación de las redes de Petri y los lenguajes formales para la descripción y síntesis de estrategias de control para sistemas autónomos.

- Definir el uso de la lógica temporal en el ámbito de la síntesis de estrategias de control para secuencias de activaciones en redes de Petri.
- Parametrizar y describir por medio de lenguajes formales y una lógica temporal, las especificaciones de activación correspondientes con el movimiento para un sistema de múltiples robots representado por una red de Petri.
- Describir e implementar un algoritmo de síntesis automática de políticas de control que interprete las descripciones dadas en lenguaje natural y lógica temporal para generar una secuencia de activaciones efectiva en tiempo continuo de una plataforma de navegación simulada con múltiples sistemas autónomos representada por una red de Petri.
- Comprobar y validar la metodología propuesta analizando y comparando los datos obtenidos (tiempo de respuesta, capacidad para encontrar una ruta solución, cantidad de transiciones) de 20 simulaciones en diversos escenarios.

Parte II

MATERIALES Y MÉTODOS

4. ESTADO DEL ARTE

La lógica temporal lineal (LTL) se ha establecido como un medio para especificar formalmente requisitos y misiones para robots y sistemas autónomos [30, 31]. Las redes de Petri se han utilizado anteriormente en diferentes problemas robóticos, por ejemplo en [32] para modelar la ejecución real del plan de movimiento, en [33] para resolver problemas de accesibilidad bajo información probabilística, o en [34] para satisfacer las tareas dadas como fórmulas booleanas.

En [29] se considera un equipo de robots idénticos y se utiliza un modelo de red de Petri (RdP) para todo el equipo lo cual brinda escalabilidad con el número de robots siempre y cuando estos sean idénticos, ya que las marcas representan a los robots y agregar más robots no cambia la topología de la RdP [33], sin embargo, en este no se tiene en cuenta la posibilidad de colisiones entre robots ya que no se tiene presente el uso de recursos compartidos. Existen

algunos trabajos como [35, 36, 37], que se pueden adaptar para evitar colisiones entre robots. En una configuración de varios robots, [38] propone un enfoque ascendente para planificar acciones, dada una especificación LTL para cada robot. En [39] se amplía el problema de enrutamiento del vehículo con restricciones de LTL y se planifica una solución basada en la Programación Lineal de Enteros Mixtos (MILP). En [31] y [40] se asume una misión única para un equipo robótico y se emplean lenguajes "trace-closed" para distribuir la misión.

Un desafío principal en la planificación de sistemas multi-robot sigue siendo la complejidad computacional. Para reducir la complejidad, en [41] se hacen resúmenes de movimientos independientes de los agentes individuales. En [42] se propone una forma de identificar partes independientes de una misión de LTL finita. En algunos trabajos se han propuesto soluciones algorítmicas para varios escenarios, como síntesis de tareas de alto nivel [27, 28], problemas de consenso [43, 44] y seguidor líder [45].

5. MARCO TEÓRICO

5.1. Marco Conceptual

Para comprender el concepto de lenguaje es necesario definir nociones más elementales como símbolo, alfabeto y palabra. Símbolo es básicamente una representación distinguible de cualquier información en una entidad indivisible; alfabeto es un conjunto no vacío de símbolos. Por ejemplo el alfabeto del idioma español sería $E = \{a, b, c, d, \ldots, z\}$ y palabra es una secuencia o cadena de caracteres construida con los símbolos de un alfabeto. De esta manera, un lenguaje es simplemente un conjunto de palabras [16].

Todos los lenguajes que comparten ciertas propiedades dadas pueden ser clasificados en conjuntos, y ya que los lenguajes en sí son conjuntos de secuencias de símbolos, se infiere que las clases de lenguajes son conjuntos de conjuntos de secuencias de símbolos [15]. N. Chomsky propuso una jerarquía de lenguajes donde las clases más complejas incluyen a las clases más simples donde se distinguen tres clases de lenguajes fundamentales en la teoría de autómatas,

los lenguajes regulares que son todos los lenguajes que se pueden formar a partir de los lenguajes básicos por medio de operaciones básicas como unión, concatenación y la cerradura de Kleene; los lenguajes libres de contexto que son lenguajes representados por un conjunto finito de variables donde cada una representa un lenguaje y los lenguajes recursivamente enumerables que son lenguajes con la capacidad de ser reconocidos por las máquinas de Turing. Este último contiene todos los anteriores.

Los lenguajes regulares o formales son llamados así debido a que sus palabras contienen regularidades o repeticiones de los mismos componentes. Un lenguaje L es regular si y solo si cumple al menos una de las siguientes condiciones: L es finito; dados dos lenguajes regulares R_1 y R_2, $L = R_1 \cup R_2$ o $L = R_1 R_2$ respectivamente; $L = R^*$, o cerradura de Kleene.

5.2. Expresiones regulares

Las expresiones regulares hacen referencia a la definición de un lenguaje en el que cada una de las palabras que lo conforman denota un lenguaje regular tal que si Σ es un alfabeto, el conjunto ER de las expresiones regulares sobre Σ contiene las cadenas en el alfabeto $\Sigma \cup \{" \wedge ", "+", "\bullet", "*", "(", ")", "\Phi"\}$ cumpliendo con: "$\wedge$" y "Φ" $\in ER$; si $\sigma \in \Sigma$, entonces $\sigma \in ER$; si $E_1, E_2 \in ER$, entonces "("E_1"$+$"E_2")" $\in ER$, "("E_1"$\bullet$"E_2")" $\in ER$, "("E_1")*" $\in ER$. Esto permite obtener una notación en la que las representaciones de los lenguajes son simplemente texto o palabras de otro lenguaje [15]. La importancia de las expresiones regulares radica en la posibilidad de expresar de forma declarativa las cadenas o palabras que se desean aceptar de determinado lenguaje [16]. De esta manera, el significado de una ER es una función $\pounds : ER \rightarrow 2^{\Sigma^*}$ que toma como entrada una expresión regular y entrega como salida un lenguaje definido como sigue:

- $\pounds("\Phi") = \emptyset$ (el conjunto vacío)
- $\pounds(" \wedge ") = \{\varepsilon\}$
- $\pounds("\sigma") = \{\sigma\}, \sigma \in \Sigma$

- $\pounds(\text{"("}R\text{"}\bullet\text{"}S\text{")"}) = \pounds(R)\pounds(S), R, S \in ER$
- $\pounds(\text{"("}R\text{"}+\text{"}S\text{"("}) = \pounds(R) \cup \pounds(S), R, S \in ER$
- $\pounds(\text{"("}R\text{")"}) = \pounds(R)^*, R \in ER$

5.3. Gramáticas formales

Una gramática es un conjunto de reglas para formar correctamente las frases de un lenguaje; la representación que se utiliza en este documento es la debida a N. Chomsky [13], y está basada en las llamadas reglas gramaticales, las cuales son una expresión de la forma $\alpha \to \beta$ en donde α y β son cadenas de símbolos en las cuales aparecen elementos del alfabeto Σ y nuevos elementos llamados *variables*. La aplicación de una regla $\alpha \to \beta$ a una palabra $\mu\alpha\upsilon$ produce la palabra $\mu\beta\upsilon$, por lo que las reglas de una gramática pueden ser vistas como reglas de reemplazo [46]. El interés se centra en las gramáticas regulares cuyas reglas son de la forma $A \to aB$ o bien $A \to a$, donde A y B son variables, y a es un caracter terminal o constante [47]. Para aplicar una gramática se parte de una variable llamada *símbolo inicial*, y se aplican repetidamente las reglas gramaticales hasta que ya no haya variables en la palabra, lo cual indica que la palabra resultante es parte del lenguaje de esa gramática [12].

Formalmente una gramática se define como una cuádrupla (V, Σ, R, S) en donde V es un alfabeto de variables; Σ es un alfabeto de constantes; R, el conjunto de reglas, es un subconjunto finito de $V \times (\Sigma V \cup \Sigma)$ y S, el símbolo inicial es un elemento de V. Para formalizar la aplicación de una gramática se debe tener en cuenta que una cadena $\mu X\upsilon$ deriva en un paso una cadena $\mu\alpha\upsilon$, escrito como $\mu X\upsilon \Rightarrow \mu\alpha\upsilon$, si hay una regla $X \to \alpha \in R$ en la gramática; una palabra $w \in \Sigma^*$ es derivable a partir de G ssi $S \Rightarrow^* w$ donde $\Rightarrow^*$ denota la cerradura reflexiva y transitiva de $\Rightarrow$. De esta manera, el lenguaje generado por una gramática G, $L(G)$, es igual al conjunto de las palabras derivables a partir de su símbolo inicial.

$$L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$$

5.4. Redes de Petri (RdP)

Una Red de Petri (RdP) es una 5-tupla $PN = \{P, T, F, W, M_0\}$ donde:

- $P = \{p_1, p_2, p_i, ..., p_m\}$ es el conjunto finito de lugares de la red.
- $T = \{t_1, t_2, t_j, ..., t_n\}$ es el conjunto finito de transiciones de la red.
- $F \subseteq (P \times T) \cup (T \times P)$ es el conjunto de arcos que definen el flujo de la red.
- $W : F \to \{1, 2, 3, \dots\}$ es la función de peso.
- $M_0 : P \to \{0, 1, 2, 3, \dots\}$ es el marcado inicial de la red.

Además, los lugares y transiciones deben cumplir que: $P \cap T = \emptyset$ y $T \cap P = \emptyset$. La estructura de una RdP sin un marcado inicial se nota por la cuádrupla $N = P, T, F, W$, y esta misma red con un marcado inicial también se puede notar como $PN = N, M_0$. En una RdP, para un lugar p_i previo a una transición t_j se dice que p_i es un lugar de entrada a la transición t_j, y además que están unidos por un arco de entrada a dicha transición. Para un lugar p_i posterior a una transición t_j, se dice que p_i es un lugar de salida de la transición t_j y además que están unidos por un arco de salida de dicha transición. El comportamiento de un sistema está representado en todo instante por el marcado actual de la red, siendo este marcado indicativo del estado y los cambios que se presentan en el sistema.

Existen múltiples configuraciones conocidas de redes de Petri, sin embargo, es de interés estudiar las aplicaciones en el modelamiento de los movimientos en sistemas robóticos. Este tipo de sistema de RdP para movimiento de robots (RMPN del inglés Robotic Motion Petri Net) es una cuádrupla $Q = (N, m_0, \prod, h)$, donde:

- $N = \langle P, T, Post, Pre \rangle$ es la estructura de la RdP con P el conjunto de lugares; T el conjunto de transiciones modelando las posibilidades de movimiento de los robots entre los lugares; $\mathbf{Post} \in \{0, 1\}^{|P| \times |T|}$ la matriz de post-incidencia definiendo los arcos de las transiciones a los lugares y $\mathbf{Pre} \in \{0, 1\}^{|P| \times |T|}$ la matriz de pre-incidencia definiendo los arcos de los lugares a las transiciones.

- $\forall t \in T, |^\bullet t| = |t^\bullet| = 1$ donde $^\bullet t$ y $t^\bullet$son el conjunto de entradas y salidas de los t lugares ($^\bullet t = \{p \in P | \mathbf{Pre}[p,t] > 0\}$ y$t^\bullet = \{p \in P | \mathbf{Post}[p,t] > 0\}$ respectivamente).
- m_0 es el marcado inicial, donde $m_0[p]$ refleja la situación del sistema al inicio.
- $\prod \cup \{\emptyset\}$ es el alfabeto de salida, donde $\emptyset$ denota una observación vacía.
- $h : P \to 2^\Pi$ es el mapa de observación, donde 2^Π es el conjunto de todos los subconjuntos de $\prod$, incluyendo el conjunto vacío $\emptyset$, y $h(p_i)$ produce la salida del lugar $p_i \in P$. Si p_i tiene al menos una marca, entonces las proposiciones de $h(p_i)$ están activas.

Para $p \in P$, el conjunto de las entradas y salidas de las transiciones se denota como $\bullet p = t \in T | \mathbf{Post}[p,t] > 0$ y $p\bullet = t \in T | Pre[p,t] > 0$, respectivamente. Una transición $t_j \in T$ está habilitada en m si todos sus lugares de entrada contienen al menos una marca, es decir, $\forall p_i \in \bullet t_j$, $m[p_i] \geq 1$. Si se dispara una transición habilitada t_j, el sistema alcanza un nuevo estado $\tilde{m} = m + C[\cdot, t_j]$, donde $C = \mathbf{Post} - \mathbf{Pre}$ es la matriz de flujo de marcas y $C[\cdot, t_j]$ es la columna correspondiente a $t_j \tilde{m}$ es una marca que se alcanza desde m al disparar t_j y se escribe como $m\langle t_j \rangle \tilde{m}$.

En una RMPN, el disparo de una transición t corresponde al movimiento de un robot desde la celda $\bullet t = p_i$ a la celda $t\bullet = p_j$. Por definición, cada transición tiene solo un lugar de entrada y solo un lugar de salida, lo que lleva a una máquina de estados [48].

Si $\widetilde{m}$ es alcanzable desde m hasta una secuencia finita de transiciones $\sigma = t_{i1} t_{i2} ... t_{ik}$, se cumple la siguiente ecuación fundamental:

$$m = m + C \cdot \sigma, |T| N \geq 0 \tag{1}$$

donde $\sigma \in \mathbb{N}_{\geq 0}{}^{|T|}$ es el vector de cómputo de disparo, es decir, su elemento j-ésimo es la cantidad acumulada de disparos de t_j en la secuencia σ. La ecuación 1 es solo una condición necesaria para la accesibilidad de una marca. Las soluciones de marcado de la ecuación 1 que no son alcanzables se llaman marcas espurias. En general, comprobar si una marca m es alcanzable

o no, no es un problema fácil debido a estas marcas espurias. Una PN es activa si desde cualquier marca alcanzable cualquier transición puede eventualmente dispararse. Además, en una máquina de estados, no existen marcas espurias [49], es decir, las soluciones de la ecuación fundamental 1 representan el conjunto de marcas alcanzables.

Dado $v_i \in 0, 1^{1\times|P|}$ como el vector fila característico de la observación $\pi_i \in \Pi$ tal que $v_i[pk] = 1$ si $\pi_i \in h(pk)$ y $v_i[pk] = 0$ en caso contrario. Es fácil observar que, para una marca alcanzable m, si el producto $v_i \cdot m > 0$ entonces la observación π_i está activa en m.

5.5. Autómatas

Los autómatas finitos corresponden a las máquinas abstractas más simples, comprendiendo como máquina abstracta simple a las abstracciones matemáticas que capturan solamente el aspecto referente a las secuencias de eventos que ocurren, sin tomar en cuenta la forma de la máquina, sus dimensiones, o si efectúa movimiento rectos o curvos. Además, los autómatas finitos se encuentran estrictamente relacionados con los lenguajes regulares [16]. Un autómata de estados finitos M es una quíntupla $(Q, \Sigma, \delta, q_0, F)$, tal que Q es un conjunto finito de estados, Σ es un alfabeto de entrada, $q_0 \in Q$ es el estado inicial, $F \subseteq Q$ es el conjunto de estados finales y δ es la función de transición que proyecta $Q \times \Sigma$ en Q. Es decir, $\delta(q, a)$ es un estado para cada estado q y cada símbolo de entrada a. Debido a que δ es una función y no solo una relación, para un estado y un símbolo del alfabeto dados, habrá un y solo un estado siguiente, lo que permite establecer que la definición dada corresponde a la de un autómata de estados finitos determinista AFD [46].

$$M = (Q, \Sigma, \delta, q_0, F)$$

Los AFD pueden ser utilizados para reconocer ciertas palabras y diferenciarlas de otras. Para que un AFD reconozca o acepte una apalabra se debe cumplir que todos los caracteres de dicha palabra sean consumidos siguiendo las transiciones y pasando de un estado a otro y que el estado al que se llega al terminar la palabra sea uno de los estados finales del autómata. Dicho

esto, se puede definir que el lenguaje aceptado por un autómata o máquina M es el conjunto de palabras aceptadas por dicha máquina, lo que en notación formal sería: una palabra $w \in \Sigma^*$ es aceptada por una máquina $M = (K, \Sigma, \delta, s, F)$ si y solo si existe un estado $q \in F$ tal que $[[s, w]] \vdash_m^* [[q, \varepsilon]]$ garantizando que lo que falta por leer al llegar a un estado final es la palabra vacía ε. $\vdash_m$ denota una relación definida formalmente como $[[q_1, \sigma w]] \vdash_m [[q_2, w]]$ para un $\sigma \in \Sigma$ si y solo si existe una transición en M tal que $\delta(q_1, \sigma) = q_2$ siendo σ el caracter que se leyó y $\vdash_m^*$ representa la cerradura reflexiva y transitiva de $\vdash_m$ [16].

En los autómatas finitos no deterministas (AFN), a diferencia de los deterministas, no se sabe exactamente cuál es la transición que se debe llevar a cabo ante un determinado evento e incluso pueden existir varias transiciones entre dos estados. Los AFD son un caso particular de los AFN, por lo que se puede decir que todo AFD es de hecho un AFN [46]. Así mismo existen los autómatas finitos con salidas que no se limitan al hecho de aceptar o no una palabra. Su aplicación se enfoca a los sistemas físicos, ya que en estos lo importante es que el sistema sea capaz de responder al entorno dependiendo del estado actual y no solo del inicial; existen dos formas de definir los autómatas finitos con salidas, el primero se conoce como los autómatas de Moore propuesto por E. Moore [50] y el segundo, como los autómatas Mealy propuesto por G. Mealy [16]. Un autómata de Moore, o máquina de Moore, es una séxtupla $M = (K, \Sigma, \Gamma, \delta, \lambda, q_0)$ donde K, Σ y δ son como en los AFD, y q_0 es el estado inicial; además se tiene Γ que es el alfabeto de salida y λ que es una función de K a Γ^*, que obtiene la salida asociada a cada estado [50] y en las máquinas de Mealy la salida depende del estado en que se encuentra el autómata y de la transición que se ejecuta, por lo tanto una máquina de Mealy es una séxtupla $M = (K, \Sigma, \Gamma.\delta, \lambda, q_0)$, en donde todos los componentes son equivalentes a los de las máquinas de Moore con excepción de λ que es una función $\lambda : K \times \Sigma \to \Gamma^*$, lo que indica que se produce una palabra formada por componentes de Γ a partir de un elemento tomado de $K \times \Sigma$ [16].

Por último es necesario definir el autómata de maniobras, también llamado planificador. Este es el encargado de tomar las decisiones, para que el autómata pueda funcionar sin necesidad de programar la totalidad de las reglas. Se encarga de verificar el estado en el que se encuentra el

autómata y dependiendo de la entrada elige y envía las reglas o instrucciones que debe utilizar el autómata para lograr un correcto desempeño. Este proceso lo realiza en instantes cortos de tiempo para disminuir el problema de explosión combinacional de estados que se presenta con la variación de los valores de verdad de los estados del autómata. A esta forma de actuar se le llama retroceso de horizonte corto [51], y consiste en que el autómata de maniobras elabora un plan para un espacio de tiempo corto y lo ejecuta constantemente, en lugar de planear una ruta larga que necesita una máquina con mayor número de estados, además de un procesador más eficiente. Debido a que el entorno cambia dinámicamente, se evalúan las entradas para espacios de tiempo cortos y se genera un autómata que cumpla las características para las entradas evaluadas en ese espacio de tiempo [52]. Este proceso se repite sucesivamente.

Autómata de Büchi

Un autómata Büchi, correspondiente a una fórmula LTL sobre el conjunto Π, tiene la estructura $B = (S, S_0, \Sigma_B, \rightarrow B, F)$, donde:

- Ses un conjunto finito de estados
- $S_0 \subseteq S$ es el conjunto de estados iniciales
- $\Sigma_B = 2^{\Pi}$ es el conjunto de entradas
- $\rightarrow_B \subseteq S \times \Sigma_B \times S$ es la relación de transición
- $F \subseteq S$ es el conjunto de estados finales

Para $s_i, s_j \in S$, $\rho(si, sj)$ es el conjunto de todas las entradas de B que permiten la transición de s_i a s_j. Las transiciones en B pueden ser no deterministas, lo que significa que desde un estado dado puede haber múltiples transiciones salientes habilitadas por la misma entrada, es decir, se puede presentar que $(s, \tau, s') \in \rightarrow_B$ y $(s, \tau, s'') \in \rightarrow_B$, con $s' \neq s''$. Por lo tanto, una secuencia de entrada puede producir más de una secuencia de estados de salida. Un autómata finito no determinista puede hacerse determinista, pero en este caso es preferible el autómata

no determinista ya que el número de estados es menor. Una palabra de entrada infinita, es decir, una secuencia de elementos de Σ_B es aceptada por B si la palabra produce al menos una secuencia de estados de B que al recorrerlos permiten visitar en el futuro un estado del conjunto F.

6. MARCO METODOLÓGICO

Dado que el cumplimiento de los objetivos específicos permite modelar el objetivo general mediante la determinación de componentes del sistema y sus relaciones, con el fin de determinar la estructura y dinámicas del sistema, se selecciona el método sistémico de investigación, dividido en fases correlacionadas con cada uno de los objetivos específicos. De esta manera, en las secciones 4 y 5 se conforma el estado del arte y se define el uso de las redes de Petri y los lenguajes formales en la descripción y síntesis de estrategias de control para sistemas autónomos.

En la sección 7.1 se hace un análisis de los tipos de lógica pertinentes para manejar y describir proposiciones atómicas cuyos valores de verdad cambian con respecto al tiempo y/o el espacio, se hace un recorrido y se ponen en evidencia algunas diferencias entre metodologías de análisis lógico de proposiciones dependientes del tiempo. En la sección 7.2 se evalúan las características definidas previamente, para con base en estas, seleccionar una lógica temporal adecuada que garantice que no se presente un problema de explosión combinacional de estados en el proceso de síntesis de las políticas de control para los sistemas de navegación autónoma.

En la sección 8 se establece formalmente la sintaxis y semántica necesaria para utilizar la lógica seleccionada en la definición del proceso de activación de una RdP abordado en la sección 8. Además, en la sección 8, se propone el problema 8.1, a través del cual se explica la necesidad de identificar tres descripciones lógicas necesarias a tener en cuenta, las cuales son: la definición y parametrización del entorno terrestre parcialmente conocido en el cual se desenvuelven los sistemas autónomos, sección 8.1, la definición y parametrización del equipo de navegación autónoma terrestre que va atener la capacidad de desenvolverse en el entorno previamente

definido, teniendo en cuenta las condiciones mínimas que debe cumplir dicho sistema para interpretar y satisfacer las características dinámicas del entorno, sección 8.2, y finalmente, la descripción a través de proposiciones atómicas y lenguajes formales de las especificaciones de activación correspondientes con el movimiento para el equipo de sistemas de navegación autónoma representado por una RdP, sección 8.3.

En la sección 9 se describe e implementa el algoritmo de síntesis de políticas de control que interpreta las descripciones previamente convertidas de lenguaje natural en forma de oraciones estructuradas a fórmulas LTL y con base en estas especificaciones genera un modelo que permite hacer abstracciones automáticas garantizando que dicho modelo permite cumplir además las especificaciones dadas para el entorno. Además se contemplan algunas de las diferentes posibilidades de configuración de solución para el problema planteado y se evalúa un caso diferente para cada una, definiendo así en la sección 9.1 el caso de un único robot ubicado en un entorno con obstáculos fijos, en la sección 9.2 el caso de un único robot ubicado en un entorno en el cual los obstáculos son móviles y por ende este debe interactuar con el entorno y en la sección 9.3 el caso de múltiples robots con una única tarea ubicados en un entorno con obstáculos fijos modelado a través de una Red de Petri.

Finalmente en la sección IV de experimentos y resultados, se plantea una serie de 20 simulaciones que permiten validar la metodología propuesta en los diferentes escenarios y se comparan los resultados obtenidos de las mismas.

La figura 1 brinda un enfoque jerárquico de la metodología adoptada partiendo desde las especificaciones dadas en lenguaje natural y describiendo cada una de las etapas que se siguen para finalmente abordar los casos de estudio propuestos en la parte IV.

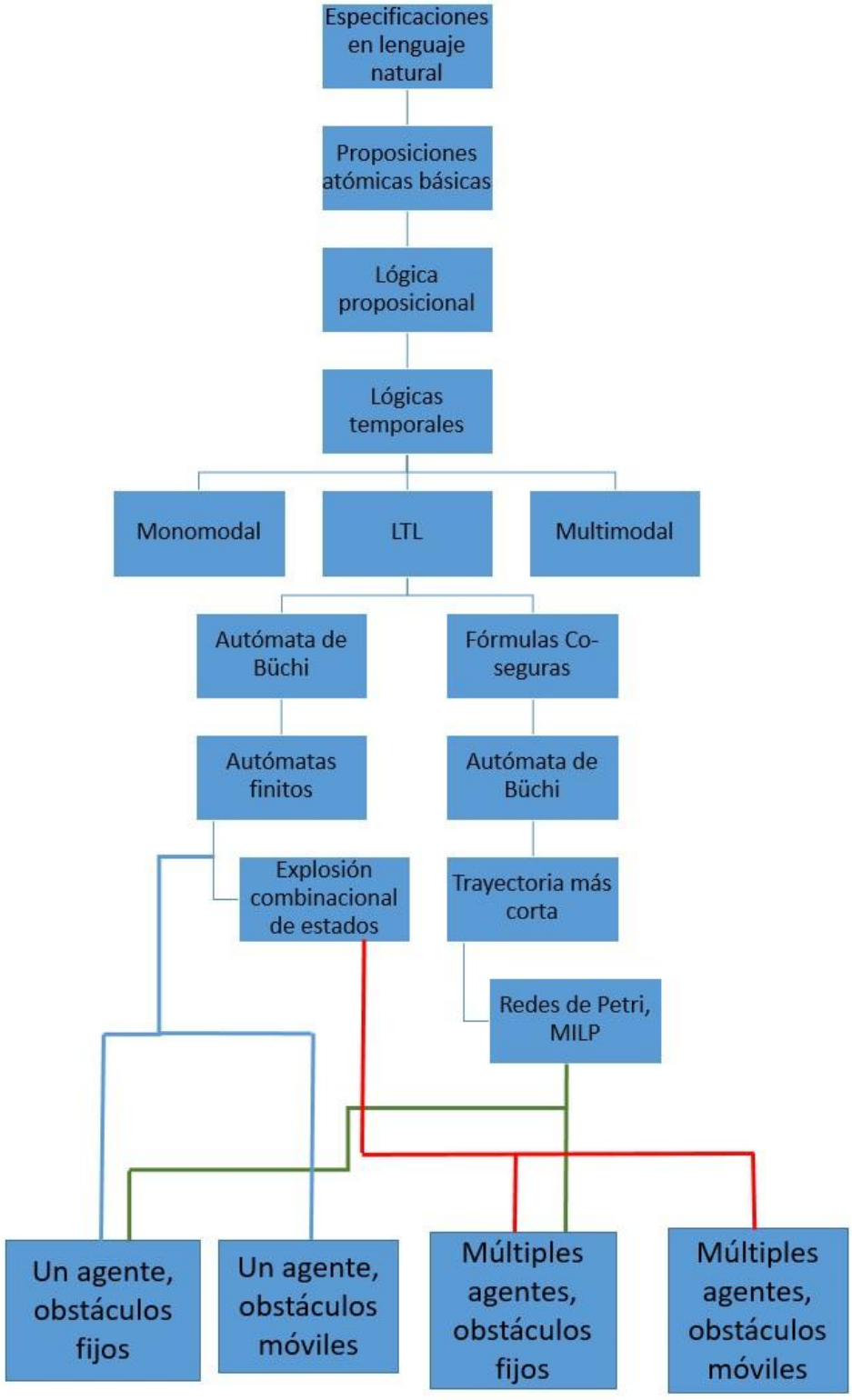

Figura 1. Diagrama metodológico

Parte III

METODOLOGÍA

7. LÓGICA TEMPORAL

7.1. Tipos de lógica para proposiciones atómicas dependientes del tiempo.

La lógica proposicional es la lógica tradicional que cuenta con los operadores conjunción $\bigwedge$, disyunción $\bigvee$, negación $\neg$, implicación $\Rightarrow$ y equivalencia $\Leftrightarrow$. La lógica proposicional permite construir un marco de referencia para la comprensión y uso del lenguaje formal, además de evaluar valores de verdad de una expresión y generar deducciones básicas. Sin embargo, la lógica proposicional no es suficiente al momento de evaluar sistemas en los que los valores de verdad de una expresión varían con respecto al tiempo, de aquí que se avance a la lógica de predicados de primer orden, la cual introduce cuantificadores universales como el operador "para todos $\forall$" y cuantificadores existenciales como el operador "existe $\exists$" permitiendo analizar formalmente frases como "un ingeniero será presidente" la cual podría expresarse de las siguientes maneras:

- $\exists$x(Ingeniero(x) & F Presidente(x)) Implica que en este momento existe alguien que es ingeniero y será presidente en algún tiempo futuro.

- $\exists$xF(Ingeniero(x) & Presidente(x)) Implica que en este momento existe alguien que en algún tiempo futuro será ingeniero y presidente.

- F$\exists$x(Ingeniero(x) & F Presidente(x)) Implica que en algún tiempo futuro va a existir alguien que primero será ingeniero y después será presidente.

- F$\exists$x(Ingeniero(x) & Presidente(x)) Implica que en algún tiempo futuro va a existir alguien que será ingeniero y presidente al mismo tiempo.

Aunque la lógica de predicados de primer orden permite analizar con un lenguaje formal expresiones más complejas en la que los valores de verdad de las proposiciones atómicas cambian con respecto al tiempo, la complejidad en la síntesis de las soluciones sobre sistemas digitales hizo que los problemas a los que se requiere dar una solución discreta sean tratados bajo las lógicas modales proposicionales, las cuales añaden uno o más pares de operadores modales resultando en sintaxis mas sencillas que las de los lenguajes de primer orden y caracterizándose por una evaluación local (en cada nodo) de las expresiones o proposiciones atómicas. Sobre estas lógicas trabajan los sistemas de transiciones etiquetados, los autómatas y otros. La lógica modal proposicional permite proponer sistemas lógicos específicos para ciertas actividades que se pueden traducir sistemáticamente a fórmulas de primer orden y además se definen con una sintaxis sencilla. En esta clase de lógica se identifican la lógica monomodal y la lógica polimodal. La primera se construye sobre el lenguaje de la lógica de proposiciones con la incorporación de dos nuevos operadores modales ($\Box, \Diamond$) y la validez de la fórmula depende de los nodos que sean visibles desde el nodo en el que se está evaluando. Cada nodo se asume como un mundo cerrado e interconectado y se supone que el comportamiento en uno de los nodos depende del conocimiento del estado de otros nodos.

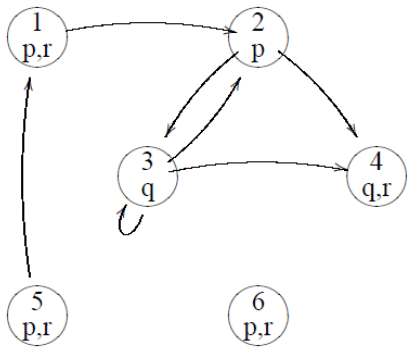

Figura 2. Ejemplo de autómata para explicación de lógicas modales.

En la lógica monomodal, el operador $\Diamond$ tiene un sentido existencial y representa la posibilidad de ocurrencia de una proposición, el operador $\Box$ tiene un sentido universal y representa la necesidad de ocurrencia de una proposición. Si se analizan las especificaciones ($p \bigwedge r$) y $\Diamond p$ para

el universo descrito por la figura 2, se tiene que la primera especificación solo sería cierta al evaluarse en el nodo 1 y que la segunda especificación sería cierta al evaluarse, por ejemplo, en el nodo 3 ya que existe al menos un nodo (el nodo 2) al que este se encuentra interconectado en el cual se cumple q .

En la lógica polimodal se introducen otros operadores modales con sus respectivos operadores duales y se puede clasificar la lógica temporal básica o bimodal que involucra como operadores el pasado y el futuro con sus respectivos operadores duales y la lógica epistémica que es muy utilizada en la economía, los juegos, los protocolos de comunicación o la cooperación entre agentes. En la lógica epistémica se plantea que no solo es necesario saber si se verifica cierta relación lógica, también interesa precisar si determinado nodo es consciente de la verificación. Para la lógica temporal bimodal simplemente se entiende la relación entre nodos como relación de precedencia entre instantes, sin embargo, existen relaciones entre instantes que no se pueden formalizar con la lógica temporal bimodal, siendo necesario introducir otros operadores modales como "Next" y "Until", dando paso así a la lógica temporal lineal (LTL).

Existe también el árbol de computación lógica (CTL) que utiliza operadores modales para considerar cada una de las ramas futuras de cada nodo, para así evaluar fórmulas del estilo "en toda rama futura existe un estado tal que" o "en alguna de las ramas futuras existe un estado tal que".

7.2. Lógica adecuada para síntesis de políticas de control en sistemas de navegación autónoma.

De las diferentes lógicas disponibles se decidió utilizar LTL ya que es un poderoso lenguaje de especificación que sirve para expresar consistentemente y sin ambigüedades un amplio rango de propiedades de sistemas [53, 54] a través de proposiciones lógicas, de operadores Booleanos estándar y de algunos operadores temporales alcanzando características que le permiten ser utilizada para razonar en cuanto al cambio con respecto al tiempo de los valores de verdad de las proposiciones atómicas (AP) y describir una gran variedad de movimientos complejos e interesantes de sistemas robóticos, con la particularidad de que las fórmulas LTL pueden

ser sintetizadas fácilmente en controladores [30]. Las fórmulas LTL se construyen a partir de proposiciones atómicas $\pi \in AP$. Este conjunto (AP) está conformado por las proposiciones atómicas previamente agrupadas en Y que representa las proposiciones del robot y X que representa las proposiciones de sensado, teniendo entonces que $AP = X \bigcup Y$. La sintaxis para construir las fórmula π puede ser definida de manera recursiva de acuerdo a la siguiente gramática [55]:

$$\varphi ::= \pi | \neg\varphi | \varphi \vee \varphi | \circ \varphi | \varphi \mu \varphi$$

Partiendo de la gramática anterior se sabe que las constantes Booleanas True y False se definen como $True = (\varphi \bigvee \neg\varphi)$ y $False = \neg True$. A partir de la negación (¬) y la disyunción ($\bigvee$) se puede definir la conjunción ($\bigwedge$) la implicación ($\Rightarrow$) y la equivalencia ($\Leftrightarrow$). Además al contar en la gramática con los operadores temporales "siguiente" ($\bigcirc$) y "hasta" (U) se pueden hallar operadores temporales adicionales como "Eventualmente" ($\Diamond\varphi = TrueU\varphi$) y "siempre" ($\Box\varphi = \neg\Diamond\neg\varphi$).

La semántica de una fórmula φ de LTL se define sobre una secuencia infinita σ de asignaciones de verdad a las proposiciones atómicas $\pi \in AP$. $\sigma(i)$ es el conjunto de proposiciones atómicas que son verdad en la posición i. A continuación se define recursivamente $\sigma, i \models \varphi$, es decir, si la secuencia σ satisface la fórmula LTL φ en la posición i.

Tabla 1. Definición recursiva de la semántica para fórmulas LTL

Relación	Definición
$(\sigma, i \models \pi)$	SI $(\pi \in \sigma(i))$
$(\sigma, i \models \neg\varphi)$	SI $(\sigma, i \neg\models \varphi)$
$(\sigma, i \models \varphi 1 \bigvee \varphi 2)$	SI $(\sigma, i \models \varphi 1)$ ó $(\sigma, i \models \varphi 2)$
$(\sigma, i \models \bigcirc \varphi)$	SI $(\sigma, i+1 \models \varphi)$
$(\sigma, i \models \varphi 1 U \varphi 2)$	SI existe en el futuro un $(k \geq i)$ tal que $(\sigma, k \models \varphi 2)$, y para todo $(i \leq j \leq k)$ existe $(\sigma, j \models \varphi 1)$

La fórmula $\bigcirc\varphi$ expresa que φ es verdad en la siguiente posición de la secuencia (el siguiente estado de tiempo) y la fórmula $\varphi 1 U \varphi 2$ expresa que $\varphi 1$ es verdad hasta que $\varphi 2$ comience a ser verdad. La secuencia σ satisface la fórmula φ si $(\sigma, 0 \models \varphi)$. La secuencia σ satisface la fórmula

$\square\varphi$ si φ es verdad en todas las posiciones de la secuencia y satisface la fórmula $\Diamond\varphi$ si φ es verdad en alguna posición de la secuencia. La secuencia σ satisface la fórmula $\square\Diamond\varphi$ si en cualquier posición φ se convierte en verdad, es decir, φ comienza a ser verdad de manera frecuente infinitamente. Para una definición formal de LTL se sugiere leer [53].

Algunas de las propiedades que se pueden expresar utilizando LTL son:

- Alcance un objetivo mientras evade obstáculos: $(\pi_1 \bigvee \pi_2 \bigvee ... \bigvee \pi_n)U\pi$ captura la propiedad de que eventualmente π va a ser cierto y hasta que eso suceda, se deben evadir los obstáculos etiquetados como $\pi_i, i = 1, ..., n$.
- Secuenciación: $\Diamond(\pi 1 \bigwedge \Diamond(\pi 2 \bigwedge \Diamond \pi 3))$ captura el requerimiento de que el robot visite primero la región π_1, luego la región π_2 y luego la región π_3, respetando ese orden.
- Cobertura: $\Diamond\pi_1 \bigwedge \Diamond\pi_2 \bigwedge ... \bigwedge \Diamond\pi_m$ especifica que el robot eventualmente alcanzará π_1 , eventualmente alcanzará $\pi_2, ...,$ eventualmente alcanzará π_m . El robot en algún momento visitará todas las regiones de interés en cualquier orden.

Clase especial de fórmulas LTL

Teniendo en cuenta que el conjunto de proposiciones atómicas AP está conformado por las proposiciones de sensado $X = x_1, x_2, ..., x_m$ y las proposiciones del robot $Y = r_1, r_2, ..., r_n, a_1, a_2, ..., a_k$ se procede a definir una clase especial de fórmulas temporales, las cuales son fórmulas LTL con la estructura $\varphi = (\varphi_e \Rightarrow \varphi_s)$ donde φ_e es una suposición acerca de las proposiciones de sensado y así, acerca del comportamiento del entorno, y φ_s representa el comportamiento deseado del robot. La fórmula φ es verdad si φ_s es verdad, es decir, si se logra el comportamiento deseado del robot, o si φ_e es falso, lo que significaría que el entorno no se comportó como se esperaba. Si se presenta que φ_e es falso, no puede garantizarse el comportamiento del sistema. Las fórmulas φ_e y φ_s tienen la siguiente estructura:

$$\varphi_e = \varphi_i^e \bigwedge \varphi_t^e \bigwedge \varphi_g^e;\ \varphi s = \varphi_i^s \bigwedge \varphi_t^s \bigwedge \varphi_g^s \tag{2}$$

Donde φ_i^e y φ_i^s son fórmulas Booleanas no temporales que restringen los valores iniciales para las proposiciones de sensado X y para las proposiciones del robot Y. φ_t^e representa la posible evolución del estado del entorno por lo cual está conformada por una conjunción de fórmulas de la forma B_i donde cada uno de los B_i es una fórmula Booleana construida de subfórmulas en $X \bigcup Y \bigcup \bigcirc X$, donde $\bigcirc X = x_1, x_2, ..., x_n$. φ_t^s representa la posible evolución del estado del robot y está compuesta por una conjunción de fórmulas con estructura B_i donde cada uno de los B_i es una fórmula Booleana en $X \bigcup Y \bigcup \bigcirc X \bigcup \bigcirc Y$. Esta fórmula restringe los valores del estado siguiente del robot $\bigcirc Y$ basado en el estado actual del robot Y y en los estados actual y siguiente de sensado $X \bigcup \bigcirc X$. Si se asume que el robot primero sensa y luego actúa entonces los valores siguientes de las proposiciones del robot pueden depender de los valores siguientes de sensado. φ_g^e, φ_g^s representan las suposiciones de objetivos para el entorno y las especificaciones de los objetivos deseados para el robot; ambas fórmulas están compuestas por una conjunción de fórmulas con estructura B_i donde cada uno de los B_i es una fórmula Booleana en $X \bigcup Y$. La clase de fórmulas LTL con estructura $\varphi = (\varphi_g^e \rightarrow \varphi_g^s)$ se denominan fórmulas de reactividad generalizada (1). Esta clase de fórmulas LTL impone restricciones adicionales sobre la estructura de las fórmulas permitidas, por lo que algunas fórmulas LTL no pueden ser expresadas, por ejemplo $\Diamond\Box\varphi$, la cual es verdad si en algún punto desconocido en el futuro φ comienza a ser verdad y permanece así por un tiempo infinito. Sin embargo, las restricciones que impone esta clase de fórmulas no implican una pérdida significativa en la expresividad ya que la mayoría de las especificaciones que se encuentran pueden ser expresadas directamente o traducidas a este formato. En otros términos, la referencia [55] declara que cualquier comportamiento que pueda ser capturado por una implicación entre conjunciones de autómatas deterministas de Büchi puede ser especificada de esta manera.

Otra clase particular de fórmulas LTL se conoce como fórmulas sintácticamente co-seguras [56]. Cualquier solución que satisfaga una fórmula LTL co-segura consiste en una cadena finita conocida como un buen prefijo, seguida de cualquier continuación infinita de proposiciones, garantizando que esta continuación no afecta el valor de verdad de la fórmula. Se demostró en [56] que cualquier fórmula LTL que contenga únicamente operadores temporales $\Diamond$ y U

cuando se escribe en forma normal positiva, es decir, cuando la negación $\neg$ aparece únicamente delante de proposiciones atómicas, es sintácticamente segura. Para este caso de fórmulas LTL, el autómata de Büchi B definido en la sección 5.5, acepta una palabra de entrada si comienza con un buen prefijo finito que lleva a B al conjunto de estados finales (la continuación del prefijo es irrelevante). Por lo tanto, la satisfacción de las fórmulas LTL co-seguras se decide basándose en las ejecuciones finitas del modelo RMPN Q definido en la sección 5.4.

8. ESPECIFICACIONES DE ACTIVACIÓN PARA UNA RdP USANDO LTL

8.1. Entornos parcialmente conocidos.

Para facilitar la explicación de los pasos que se llevaron a cabo en esta metodología, se plantea el desarrollo del siguiente problema:

Problema 1: se considera un robot o equipo de robots que se mueven en un entorno cuadrado con 5 áreas de interés denotadas por a_1, a_2, a_3, a_4, a_5 como el que se muestra en la figura 3, donde las líneas negras tienen correspondencia física en el espacio de trabajo y las líneas amarillas representan el particionamiento propuesto. El comportamiento deseado para el sistema dado en lenguaje natural es: "eventualmente visite las regiones a_1, a_2, a_3 y $a5$ evitando siempre la región a_4 y repita esta instrucción de manera infinita".

Una solución propuesta para este "problema de planificación del movimiento" consiste en hacer una abstracción del comportamiento del robot para así particionar el espacio de trabajo obteniendo un espacio fragmentado en regiones, como el mostrado en la figura 3, y así poder generar una fórmula LTL adecuada que permita describir las trayectorias posibles del robot y que pueda ser sintetizada en un autómata [57].

El segmento de fórmula LTL ligado con la morfología y el particionamiento del espacio de trabajo puede expresarse para el problema en cuestión como:

$$a_1 \Rightarrow (\bigcirc a_1 \bigvee \bigcirc a_2 \bigvee \bigcirc a_3 \bigvee \bigcirc a_4) \bigwedge a_2 \Rightarrow (\bigcirc a_1 \bigvee \bigcirc a_2 \bigvee \bigcirc a_4 \bigvee \bigcirc a_5) \ldots \bigwedge \Box(a_5 \Rightarrow (\bigcirc a_2 \bigvee \bigcirc a_3 \bigvee \bigcirc a_4 \bigvee \bigcirc a_5)) \quad (3)$$

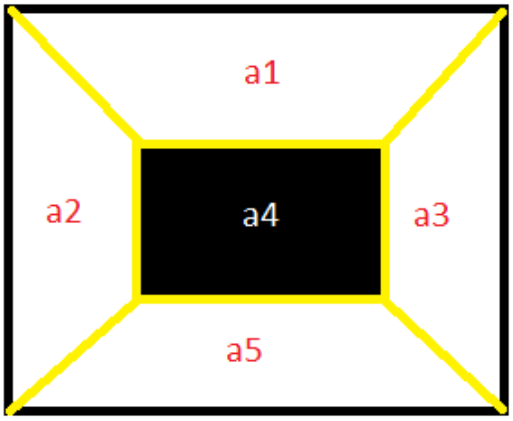

Figura 3. Espacio de trabajo propuesto para el problema a solucionar.

La fórmula 3 define la adyacencia entre regiones, en donde se le indica al robot que al encontrarse en una región solo podrá avanzar a una región adyacente especificada. Esta última fórmula se puede generar de manera automática a través de un programa de fácil implementación.

Otro enfoque que permite solucionar el problema propuesto en términos de la definición y parametrización del entorno en que se desenvuelve un sistema de navegación autónoma terrestre, implica el uso de una red de petri para movimiento de robots (RMPN) 5.4. Se considera el mapa del entorno mostrado en la figura 3 que consta de 5 celdas $a_1, ..., a_5$, un robot localizado inicialmente en $a1$ y cinco regiones de interés $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ de modo que la región π_1 corresponde a la celda a_1 y al lugar p_1, la región π_2 corresponde a a_2 y al lugar p_2 y así respectivamente con todas las celdas y regiones. El modelo RMPN para el entorno descrito previamente se puede apreciar en la figura 4, en la cual, $P = p_1, ..., p_6 \text{y} T = t_1, ..., t_{16}$. Como el conjunto de transiciones de entrada de p_1 es $\bullet p1 = t_2, t_4, t_6$ entonces $Post[p_1, t_2] = Post[p_1, t_4] = Post[p_1, t_6] = 1$ mientras $Post[p_1, t_j] = 0$ para todos $t_j \in T \backslash \bullet p1$. Además, dado que el conjunto de transiciones de salida de p_1 es $p_1\bullet = t_1, t_3, t_5$ entonces $Pre[p_1, t_1] = Post[p_1, t_5] = Post[p_1, t_7] = 1$, mientras $Pre[p_1, t_j] = 0$ para todo $t_j \in T \backslash p1\bullet$.

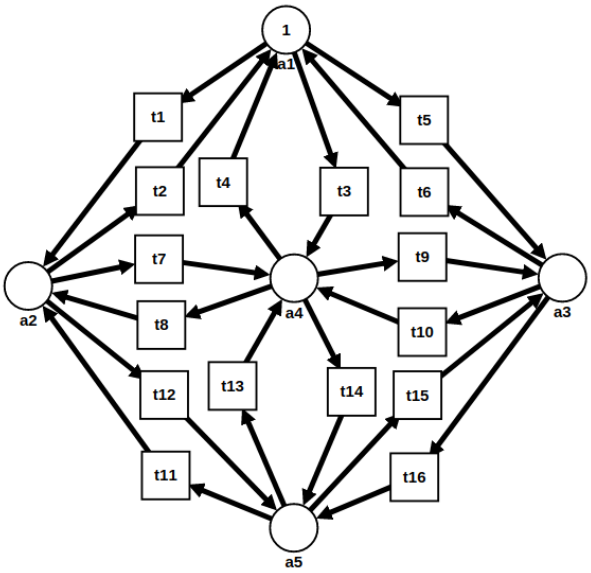

Figura 4. Modelo RMPN para el espacio de trabajo propuesto en 8.1

8.2. Metodología para definir y parametrizar un equipo de sistemas de navegación autónoma terrestre

El equipo de navegación autónoma terrestre a parametrizar debe contar con la capacidad de desenvolverse en un entorno previamente definido, teniendo en cuenta las condiciones mínimas que debe cumplir para interpretar y satisfacer las características dinámicas del entorno.

Se asume que el tipo de robots móviles utilizados opera en un espacio de trabajo poligonal P. El movimiento del robot está expresado por:

$$\dot{p}(t) = u(t) \quad p(t) \in P \subseteq \mathbb{R}^2 \quad u(t) \in U \subseteq \mathbb{R}^2 \tag{4}$$

Donde $p(t)$ es la posición del robot en el tiempo t y $u(t)$ es la entrada de control. Se asumirá que el espacio de trabajo P está particionado en un número finito de celdas $P1, ..., Pn$, donde $P = \bigcup_{i=1}^{n} P_i$ y $P_i \bigcap P_j = \varnothing$ si $i \neq j$. Además se considera cada una de las celdas como un polígono convexo. Al particionar el espacio de trabajo se genera una serie de proposiciones Booleanas $r_1, r_2, ..., r_n$ las cuales son ciertas si el robot se encuentra en la posición P_i.

$$r_{i=}\begin{cases} True & \text{Si} \quad p \in P_i \\ False & \text{Si} \quad p \notin P_i \end{cases}$$

Debido a que P_i es una partición de P, solamente un r_i puede ser cierto en cualquier momento. Adicional a esto, en este caso el robot puede ejecutar en cualquier momento acciones extras como encender una luz. Esto implica que estas acciones también deben estar codificadas como proposiciones atómicas. Para este conjunto de proposiciones $A = a_1, a_2, ..., a_k$ se tiene:

$$a_{i=}\begin{cases} True & \text{Cuando Si se ejecuta la acción} \quad i \\ False & \text{Cuando No se ejecuta la acción} \quad i \end{cases}$$

Se define entonces $act(t) \subseteq A$ como el conjunto de acciones que están activas o que son verdad en un tiempo t, es decir: $a_i \in act(t)$ SI a_i es verdad en un tiempo t. Entonces el conjunto Y que define todas las proposiciones asociadas al robot está dado por la unión de los conjuntos que representan las proposiciones de las acciones y las proposiciones de localización en el espacio de trabajo, esto es $Y = P \bigcup A = r_1, r_2, ..., r_n, a_1, a_2, ..., a_k$. Si dos acciones de este conjunto no pueden ser ciertas al mismo tiempo, se debe agregar a las especificaciones del sistema.

8.3. Modelado de un sistema de navegación autónoma con Redes de Petri.

Para facilitar la identificación de los pasos a desarrollar en la metodología se asume un equipo de robots R idénticos modelados bajo los parámetros de la sección 8.2 que se mueven en un entorno rectangular. El entorno se divide en un conjunto finito de células disyuntas como el mostrado en la figura 5, utilizando, por ejemplo, un enfoque de descomposición celular [6], y se ubican 2 robots representados por los círculos verdes, encargados de cumplir de manera conjunta las tareas asignadas. Una celda del entorno se denota como $a_j, , j = 1, ..., |A|$ donde A es el conjunto de todas las celdas y $|A|$ denota la cardinalidad del conjunto A. Además, se asume un conjunto finito de proposiciones atómicas $\Pi = \pi 1, \pi 2, ..., \pi|\Pi|$, donde, π_i etiqueta una región específica de interés. Una región de interés π_i corresponde a una o más celdas del

entorno, y si al menos un robot se encuentra en cualquiera de estas celdas, se dice que la proposición π_i se cumple ($True$). El conjunto Π se utiliza para proporcionar una fórmula LTL que defina la tarea que debe cumplir el equipo de robots.

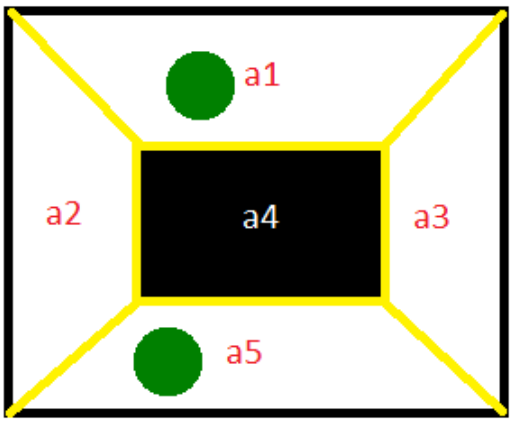

Figura 5. Espacio de trabajo con un equipo de robots

Para el nuevo problema, la RMPN obtenida se puede apreciar en la figura 6 en la cual los robots se representan como marcas verdes en los lugares a_1 y a_5 de la red de Petri. El marcado inicial del sistema de red es $m_0 = [1, 0, 0, 0, 1]^T$, el alfabeto de salida es $\Pi = \pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ y el mapa de observación: $h(a_1) = \pi_1, h(a_2) = \pi_2, h(a_3) = \pi_3, h(a_4) = \pi_4 \text{y } h(a_5) = \{\pi_5\}$. El vector característico de π_1 es $v1 = [1, 0, 0, 0, 0]$ ya que π_1 se puede observar solo en a_1, $v2 = [0, 1, 0, 0, 0]$ ya que π_2 se puede observar solo en a_2, $v3 = [0, 0, 1, 0, 0]$ ya que π_3 se puede observar solo en a_3, y $v5 = [0, 0, 0, 0, 1]$ ya que π_5 se puede observar solo en a_5. Con los anteriores vectores se construlle la matriz de transiciones así:

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

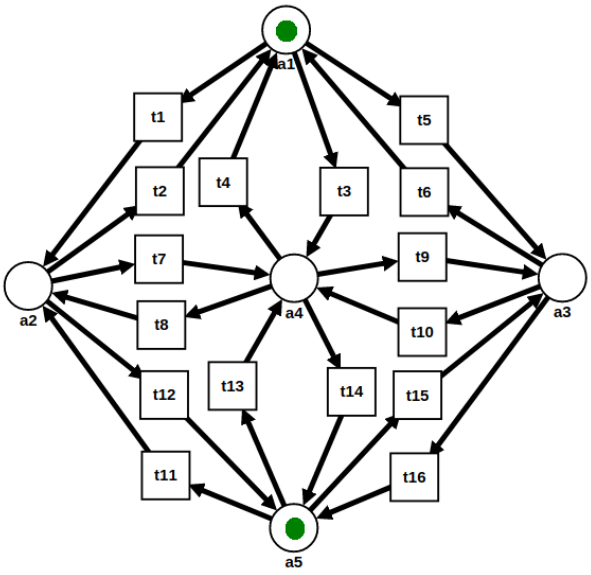

Figura 6. RMPN para 2 robots en el espacio detrabajo de la figura 5.

Debido a que $V \cdot m_0 = [1, 0, 0, 1^T]$, la observación π_1 y π_5 están activas en m_0 debido a que la ubicación inicial de los robots es en a_1 y a_5.

Una ejecución (o trayectoria) de Q es una secuencia finita $r = m_0[t_{j_1}\rangle m_1[t_{j_2}\rangle m_2[t_{j_3}...t_{j_{|r|}}\rangle m_{|r|}$ que produce una palabra de salida, que es la secuencia observada de elementos de 2^{Π} y las fórmulas LTL se interpretan sobre cadenas infinitas de observaciones desde 2Π [58]. Como en este caso se consideran solo las fórmulas LTL co-seguras definidas en la sección 7.2, cualquier fórmula LTL sobre el conjunto Π se puede transformar en un autómata de Büchi 5.5, que acepta solo las cadenas de entrada que satisfacen la fórmula [59]. Algunas herramientas de software disponibles que permiten tales conversiones son [60, 61].

Teniendo en cuenta que las especificaciones de activación para la RdP se van a generara con base en un requerimiento dado como una fórmula Booleana, se define un conjunto finito de proposiciones atómicas $\prod = \{\prod_1, \prod_2, \prod_3, ..., \prod_{|\prod|}\}$, donde las etiquetas $\prod_i$ representan una región de interés específica del entorno. Los requerimientos se expresan como una fórmula lógica Booleana sobre el conjunto de variables $\mathscr{P} = \mathscr{P}_t \cup \mathscr{P}_f$, donde $\mathscr{P}_t = \prod$ y $\mathscr{P}_f = \{\pi_1, \pi_2, ..., \pi_{|\prod|}\}$. Los conjuntos $\mathscr{P}_t$ y $\mathscr{P}_f$ hacen referencia a las mismas regiones de interés, pero los elementos de $\mathscr{P}_t$ indican las regiones que deben ser visitadas a lo largo de la ejecución

de la trayectoria y los elementos de $\mathscr{P}_f$ indican las regiones que deben ser visitadas en el último estado de la ejecución. Las especificaciones son interpretadas como palabras finitas sobre el conjunto 2^{Π}, igual que la RdP con salidas definida en la sección 5.4. En general, la composición del conjunto $\mathscr{P}$ se evalúa sobre la palabra generada por la ejecución $r = m_0[t_{j_1}\rangle m_1[t_{j_2}\rangle m_2[t_{j_3}...t_{j_{|r|}}\rangle m_{|r|}$ teniendo en cuenta las siguientes condiciones:

- $\prod_i \in \mathscr{P}_t$ es verdadero al evaluarlo sobre la palabra $h(r)$ si y solo si $\exists j \in \{0, 1, ..., |r|\}$ tal que $\prod_i \in ||V \bullet m_j||$.
- $\prod_i \in \mathscr{P}_f$ es verdadero al evaluarlo sobre la palabra $h(r)$ si y solo si $\prod_i \in ||V \bullet m_j||$.

Es decir, una variable mayúscula corresponde a una proposición que se evalúa en cualquier momento a lo largo de una ejecución, mientras que una variable minúscula corresponde la evaluación de una proposición únicamente en el estado final de una ejecución. Además, todos los requerimientos φ de base Booleana deben expresarse en una Forma Normal Conjuntiva (FNC) y tales requerimientos representan la tarea que debe cumplir todo el equipo de robots y no especifica las funciones de cada robot de manera individual. Todas las expresiones lógicas se pueden expresar en FNC [62] y al expresar φ en FNC, se tiene una conjunción de n términos $\varphi = \varphi_1 \wedge \varphi_2 \wedge, ..., \wedge \varphi_n$. Cada uno de los términos $\varphi_i \, i = 1, 2, ..., n$ es una disyunción de n_i variables del conjunto $\mathscr{P}$ con la forma $[\pi_2 | \neg \pi_2] \vee ... \vee [\prod_{jn_i} | \prod_{jn_i}] \vee [\pi_{jn_i} | \neg \pi_{jn_i}]$. Como cualquier fórmula Booleana de la forma FNC puede ser convertida a un conjunto de desigualdades lineales utilizando diversas técnicas [63], se define un vector binario $x = [x_{\prod_1}, x_{\prod_2}, ..., x_{\prod_{|\prod|}}, x_{\pi_1}, x_{\pi_2}, ..., x_{\pi_{|\prod|}}]^T \in \{0, 1\}^{2 \bullet \prod}$ con $2 \bullet \prod$ variables evaluando para cada componente del vector las siguientes condiciones:

- $x_{\prod_i} = 1$ si la proposición $\prod_i$ se evalúa cierta, es decir, si la región etiquetada como $\prod_i$ es visitada en cualquier momento durante la ejecución de la trayectoria y $x_{\prod_i} = 0$ de cualquier otra manera.
- $x_{\pi_i} = 1$ si la proposición $\pi_i$se evalúa cierta, es decir, si un robot se detiene dentro de región etiquetada como π_i y $x_{\pi_i} = 0$ de cualquier otra manera, $\forall i = 1, 2, ..., |\prod|$.

De esta manera, la satisfacción de la especificación φ es equivalente a un conjunto de n desigualdades lineales donde cada una corresponde a un término disyuntivo $\varphi_i, i = 1, ..., n$. Para construir formalmente estas desigualdades, para cada φ_i se define una función $\alpha_i : \mathscr{P} \to \{-1, 0, 1\}$ que muestra qué variables de $\mathscr{P}$ aparecen en la disyunción φ_i y cuales de estas son negadas.

$$\alpha_i(\gamma) = \begin{cases} -1 & Si\, \neg\gamma\, aparece\, en\, \varphi_i \\ 0 & Si\, \gamma\, no\, aparece\, en\, \varphi_i\, \forall\gamma \in \mathscr{P} \\ 1 & Si\, \gamma\, aparece\, en\, \varphi_i \end{cases} \tag{5}$$

Formalmente, la desigualdad lineal correspondiente a la disyunción φ_i está dada por:

$$\sum_{\gamma \in \mathscr{P}} (\alpha_i(\gamma) \bullet x_\gamma) \geq 1 + \sum_{\gamma \in \mathscr{P}} min(\alpha_i(\gamma), 0) \tag{6}$$

Donde $min(\alpha_i(\gamma), 0)$ es el mínimo valor entre α_i y 0. Las ecuaciones 5 y 6 parten de las siguientes suposiciones: si la región correspondiente al símbolo $\gamma \in \mathscr{P}$ no es visitada según φ_i, entonces su variable binaria correspondiente tiene un coeficiente $\alpha_i(\gamma) = 0$. De todas las regiones que aparecen no negadas en la disyunción φ_i, al menos una debe ser visitada y por lo tanto, la suma de todas sus variables binarias correspondientes debe ser mayor o igual a 1. Un símbolo γ negado significa la evasión de una región, ya sea a lo largo de una trayectoria o en el estado final, lo cual implica que su variable binaria correspondiente x_γ debería ser cero. De esta manera, una especificación de la forma FNC es algorítmicamente convertible, a través de 6 en un sistema de n desigualdades lineales, una por cada término disyuntivo. Para cada una de las observaciones de $\prod_i$ se plantea una variable binaria $x_{\pi_i} = 1$ si π_i se evalúa como verdadera en un estado final de la ejecución. En la ecuación 7 se plantea un conjunto de desigualdades lineales que puede ser utilizado para definir el valor de la variable binaria x_{π_i} en un marcado final m donde N es el número de robots y $v_{\prod_i}$ es el vector característico de observaciones de $\prod_i$.

$$\begin{cases} N \bullet x_{\pi_i} \geq v_{\prod_i} \bullet m \\ x_{\pi_i} \leq v_{\prod_i} \bullet m \end{cases} \tag{7}$$

Al encontrar una solución para el problema propuesto, el objetivo es minimizar el número de transiciones a lo largo de la trayectoria. Por lo tanto, se elige la función de costo $1^T \bullet \sigma$ y se formula el problema 8 de Programación Lineal Entera Mixta (MILP) para obtener un marcado final en el que se cumpla la especificación. Donde v_γ es el vector característico de $\gamma \in \mathscr{P}$, de aquí, la solución se obtiene activando las transiciones habilitadas y almacenando la secuencia de lugares visitados por cada marca.

$$s.t. \left| \begin{array}{c} min\, 1^T \bullet \sigma \\ m = m_0 + C \bullet \sigma \\ \sum_{\gamma \in \mathscr{P}} (\alpha_i(\gamma) \bullet x_\gamma) \geq 1 + \sum_{\gamma \in \mathscr{P}} min(\alpha_i(\gamma), 0), \forall \varphi_i \\ N \bullet x_\gamma \geq v_\gamma \bullet m, \forall \gamma \in \mathscr{P} \\ x_\gamma \leq v_\gamma \bullet m, \forall \gamma \in \mathscr{P} \\ m \in \mathbb{N}_{\geq 0}^{|P|}, \sigma \in \mathbb{N}_{\geq 0}^{|T|}, x \in \{0,1\}^{|\mathscr{P}|} \end{array} \right. \tag{8}$$

Para incluir el cumplimiento de las restricciones sobre la trayectoria, se considera una secuencia de k marcas $m_1, m_2, ..., m_k$ tal que: $m_1 = m_0 + C \bullet \sigma_1$, $m_0 - Pre \bullet \sigma_1 \geq 0$; $m_2 = m_1 + C \bullet \sigma_2$, $m_1 - Pre \bullet \sigma_2 \geq 0$; ... lo que implica que entre los estados de la RdP m_{i-1} y m_i, cada marca se mueve a lo sumo a través de una transición y así se evita el disparo de transiciones para lugares vacíos. Para cada una de las especificaciones $\prod_i$ que pertenecen a la restricción de la trayectoria se introduce una variable binaria $x_{\prod_i} = 1$ siempre y cuando se evalúe cierta a lo largo de la trayectoria y como la trayectoria está dada por la secuencia de las k marcas intermedias se define 9 como el conjunto de las desigualdades lineales que consideran todas las marcas intermedias y no solo la marca final como en 7.

$$\begin{cases} N \bullet (k+1) \bullet x_{\prod_i} \geq v_{\prod_i} \bullet (\sum_{j=0}^{k} m_j) \\ x_{\prod_i} \leq v_{\prod_i} \bullet (\sum_{j=0}^{k} m_j) \end{cases} \tag{9}$$

9. METODOLOGÍA PARA SÍNTESIS DE POLÍTICAS DE CONTROL A PARTIR DE DESCRIPCIONES DADAS EN LÓGICA TEMPORAL

Para generar un modelo que permita hacer abstracciones automáticas, garantizando que se cumplen las especificaciones dadas para el entorno parcialmente definido, es necesario diferenciar las posibles soluciones existentes a este problema de movimiento, ya que no es igual la solución para un solo robot que para un equipo de robots, así como no es igual si el robot puede interactuar con el entorno adquiriendo constantemente información de este, o si el robot solo ejecuta la tarea planeada.

9.1. Un robot y obstáculos fijos

Para el caso de un único robot modelado bajo los parámetros de la sección 8.2, ubicado en el espacio de trabajo mostrado en la figura 3, con una tarea dada en lenguaje natural como "eventualmente visite las regiones a_1, a_2, a_3 y a_5 evitando siempre la región a_4 y repita esta instrucción de manera infinita", la solución aquí argumentada consiste en generar una fórmula LTL adecuada que permita describir las trayectorias posibles del robot y que pueda ser sintetizada en un autómata. Inicialmente la fórmula LTL puede interpretarse como una conjunción de 3 condiciones que deben cumplirse siempre, las cuales son:

- La condición que describe el objetivo de visitar continuamente las regiones $a1, a2, a3$ y $a5$ evitando pasar por la región $a4$.
- La restricción de exclusión entre regiones en la que se le indica al robot que solo puede ocupar una región a la vez en el instante siguiente de tiempo.

- La especificación de adyacencia entre regiones en donde se le indica al robot que al encontrarse en una región solo podrá avanzar a una región adyacente especificada en la fórmula. Este segmento de fórmula fue hallado en la sección 8.1 y corresponde a la ecuación 3.

La primera condición descrita como una fórmula LTL es:

$$(\square\Diamond a_1)\bigwedge(\square\Diamond a_2)\bigwedge(\square\Diamond a_3)\bigwedge(\square\Diamond a_5)\bigwedge(\square\neg a_4) \tag{10}$$

En la ecuación 10, los primeros 4 términos representan las condiciones de visitar continuamente estas regiones indeterminadamente y el último término representa la condición de evitar siempre la región a_4. La siguiente restricción de exclusión entre regiones es un arreglo de fórmulas del tipo:

$$\wedge\square((\bigcirc a_1\bigwedge_{i\neq 1}\neg\bigcirc a_i)\bigvee(\bigcirc a_2\bigwedge_{i\neq 2}\neg\bigcirc a_i)\bigvee\ldots\bigvee(\bigcirc a_5\bigwedge_{i\neq 5}\neg\bigcirc a_i)) \tag{11}$$

En la ecuación 11, cada uno de los términos contenidos en los paréntesis indica que en el instante siguiente de tiempo solo puede ser cierta una condición, es decir, que el robot solo podrá ocupar un único espacio futuro lo que ayuda a disminuir el alto costo computacional al reducir considerablemente las combinaciones de estados futuros del sistema.

Finalmente la conjunción de las ecuaciones 3, 10 y 11 describe el plan de movimiento deseado para el sistema. El paso siguiente es verificar que la fórmula LTL generada sea sintetizable en un autómata de estados finitos, representado por un diagrama de estados y transiciones. La solución discreta del problema de planificación del movimiento para un único robot se presenta como un autómata donde el alfabeto de entrada corresponde a la información del entorno en que se va a desenvolver el sistema y el alfabeto de salida a los movimientos y acciones del robot. Para hacer la verificación de la fórmula LTL se pueden utilizar numerosas herramientas de chequeo de modelos como SPIN, NuSMV y otras [64]. En este caso se utilizó

SPIN, obteniendo como resultado el autómata representado por el diagrama de estados y transiciones contenido en la figura 7.

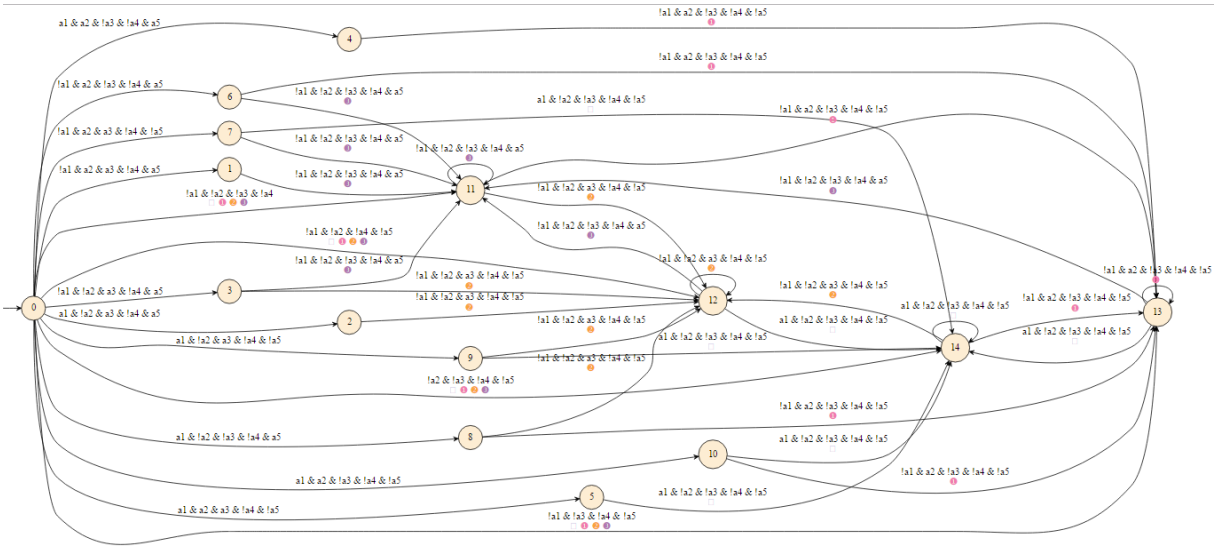

Figura 7. Autómata obtenido como solución discreta.

El plan de movimiento obtenido que se representa a través del diagrama de estados y transiciones mostrado en la figura 7 satisface las especificaciones dadas para el sistema en lenguaje natural, por lo cual se dice que la fórmula LTL que encierra las especificaciones es sintetizable en un autómata. Si el entorno se comporta de manera diferente a la esperada, por ejemplo si se presentara un traslape entre regiones, el autómata no tendría una transición adecuada para cumplir con el plan de movimiento por lo cual ya no sería válido. El autómata obtenido no es el único que puede cumplir con las especificaciones utilizando el menor número de transiciones posibles [65]. Analizando detenidamente el diagrama de estados y transiciones de la figura 7 se pueden identificar de manera clara estados que no pueden ser interpretados por un robot, ya que no se ha generado una función de etiquetado previa para estos; tal es el caso de los estados $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$. Esta particularidad se presenta debido a que en la fórmula LTL generada para describir el plan de movimiento no se tuvo en cuenta las condiciones iniciales, en este caso la región inicial donde está el robot. La omisión de esta restricción implica que en el proceso de síntesis del autómata se deben evaluar todas las posibles configuraciones partiendo desde un estado inicial incierto que puede cumplir o no con las especificaciones dadas.

Para dar solución al inconveniente expuesto, se propone incluir la condición inicial en la fórmula LTL, lo cual implica una transformación de la fórmula que antes era una sucesión de conjunciones de la forma $(\varphi = \varphi_1, \varphi_2, ..., \varphi_n)$ a una implicación de conjunciones de la forma $((\varphi_1, \varphi_2, ..., \varphi_n) \Rightarrow (\Phi_1, \Phi_2, ..., \Phi_m))$, en donde el término ubicado a la izquierda de la implicación tiene como función determinar las condiciones iniciales del sistema y el entorno y el término derecho la ejecución del plan de movimiento. La subfórmula LTL que representa la restricción de condición inicial para el problema propuesto es $(a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5)$, la cual indica que para sintetizar un plan de movimiento adecuado se debe cumplir que el robot se encuentre ubicado inicialmente en la región a_1. Al utilizar la herramienta de chequeo de modelos para sintetizar un autómata a partir de la nueva fórmula LTL, se obtiene como resultado el diagrama de estados y transiciones que se muestra en la figura 8, en el cual, además de la evidente diferencia en la cantidad de estados y transiciones con el de la figura 7, se nota que los estados de indeterminación del robot desaparecen. Esto se debe a que se indicó que si se cumple la condición inicial entonces se sintetiza un autómata para el plan de movimiento. También puede identificarse en el diagrama que en caso de que no se cumpla la condición inicial, es decir, que al comenzar el robot se encuentre en una región diferente a a_1, el autómata se irá a un estado absorbente que indica error. Siguiendo esta metodología pueden generarse incluso planes de movimiento que indiquen secuenciación aparte de cobertura.

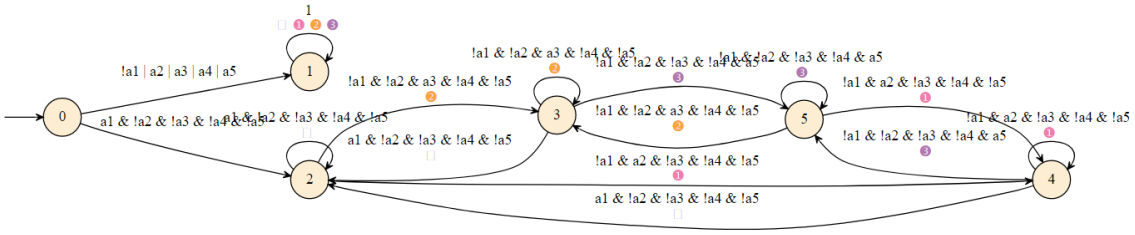

Figura 8. Diagrama de estados y transiciones para fórmula LTL con condiciones iniciales.

Se plantea entonces el algoritmo 1 que recoge las técnicas expuestas anteriormente y que permite realizar síntesis continua de políticas de control para sistemas con un único robot que se desenvuelve en un entorno con obstáculos fijos.

Algorithm 1 Síntesis continua de políticas de control para la sección 9.1

Require: Entorno, Regionesinteres, especificacionLTL
Ensure: trayectoria

```
P ← GetPartTriang (Entorno, Regionesinteres)
AutEntorno ← GetAutEnt(P)
if AutEntorno = false then
    return Error al definir el entorno
end if
AutBuchi ← GetAutBuchi(especificacionLTL)
if AutBuchi = false then
    return Error en formula LTL
end if
AutGeneral ← GetAutGeneral(AutEntorno, AutBuchi)
trayectoria ← findRunAcep(AutGeneral)
return trayectoria
```

9.2. Un robot y obstáculos móviles

Para el caso de un único robot modelado bajo los parámetros de la sección 8.2, ubicado en el espacio de trabajo R mostrado en la figura 9, que se encuentra particionado en 16 regiones las cuales se relacionan con las proposiciones de localización del robot $R = r1, r2, ..., r16$, inicialmente ubicado en la región 2 y con una especificación dada en lenguaje natural como "Diríjase de la región 2 a la región 15 y luego regrese a la región 2. Si en el camino se encuentra un obstáculo, encienda una luz y permanezca en el mismo lugar hasta que el obstáculo desaparezca. Si el obstáculo desaparece, apague la luz y retome su camino".

Como los obstáculos hacen parte del entorno, el conjunto de proposiciones de sensado contiene solo una proposición $X = S^{obs}$ la cual se vuelve cierta si el robot detecta un obstáculo. Las suposiciones sobre los obstáculos son capturadas por $\varphi_e = \varphi_i^e \bigwedge \varphi_t^e \bigwedge \varphi_g^e$. El robot inicialmente no detecta ningún obstáculo, por lo tanto $\varphi_i^e = (\neg S^{obs})$. Se asume que el robot solo puede detectar obstáculos en regiones diferentes a la región 2 y la región 15, por lo que se codifica la restricción de que en las regiones 2 y 15 el valor de S^{obs} no puede cambiar. Este requerimiento es capturado por la fórmula: $\varphi_t^e = \square((\neg r_1 \bigwedge \neg r_3 \bigwedge ... \bigwedge \neg r_{13} \bigwedge \neg r_{14} \bigwedge \neg r_{16}) \Rightarrow (\bigcirc S^{obs} \Leftrightarrow S^{obs}))$. Como no se suponen más hipótesis sobre las proposiciones ambientales, entonces $\varphi_g^e = (True)$. Ahora, para modelar el robot y las especificaciones deseadas que son capturadas por $\varphi_s =$

$\varphi_i^s \bigwedge \varphi_t^s \bigwedge \varphi_g^s$ se definen 17 proposiciones del robot expresadas en $Y = r_1, r_2, ..., r_{16}, a^{luzOn}$. Como condición inicial de localización, el robot podría comenzar en la región 2 o en la región 15 con la luz apagada ya que en estas regiones no deben haber obstáculos. Se tiene entonces la ecuación 12:

$$\varphi_i^s = \begin{cases} (r_2 \wedge_i \in \{1, 3, ..., 16\} \neg r_i \wedge \neg a^{luzOn}) \\ \vee (r_{15} \wedge_i \in \{1, ..., 13, 14, 16\} \neg r_i \wedge \neg a^{luzOn}) \end{cases} \tag{12}$$

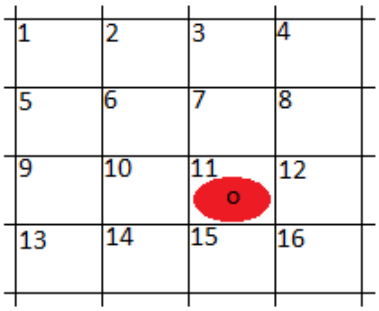

Figura 9. Espacio de trabajo para un robot que interactúa con el ambiente

La fórmula φ_t^s modela los posibles cambios en el estado del robot. El primer bloque de subfórmulas que la componen representa las posibles transiciones entre las regiones, por ejemplo, de la región 1 el robot podría desplazarse a la región 2 o a la región 5 o podría permanecer en la región 1. La siguiente subfórmula representa la restricción de mutua exclusión entre regiones que especifica que en cualquier momento sólo puede ser cierta una región de R. El último bloque de subfórmulas representa las especificaciones deseadas para el sistema y establece que si el robot se encuentra un obstáculo debe permanecer inmóvil con la luz encendida hasta que este desaparezca. Además, especifica que mientras el robot no encuentre un obstáculo, la luz debe estar apagada. Se define φ_t^s en la ecuación 13:

$$\varphi_t^s = \begin{cases} \begin{cases} \wedge\Box(r_1 \Rightarrow (\bigcirc r_1 \vee \bigcirc r_2 \vee \bigcirc r_5)) \\ \wedge\Box(r_2 \Rightarrow (\bigcirc r_2 \vee \bigcirc r_1 \vee \bigcirc r_3 \vee \bigcirc r_6)) \\ \vdots \\ \wedge\Box(r_{16} \Rightarrow (\bigcirc r_{16} \vee \bigcirc r_{15} \vee \bigcirc r_{12})) \end{cases} \\ \wedge\Box((\bigcirc r_1 \wedge_{i\neq 1} \neg \bigcirc r_i) \\ \vee(\bigcirc r_2 \wedge_{i\neq 2} \neg \bigcirc r_i) \\ \vdots \\ \vee(\bigcirc r_{16} \wedge_{i\neq 16} \neg \bigcirc r_i)) \\ \begin{cases} \wedge\Box(\bigcirc S^{obs} \Rightarrow (\wedge_{i\in\{1,2,\ldots,16\}} \bigcirc r_i \Leftrightarrow r_i) \wedge \bigcirc a^{luzOn}) \\ \wedge\Box(\neg \bigcirc S^{obs} \Rightarrow \neg \bigcirc a^{luzOn}) \end{cases} \end{cases} \tag{13}$$

Finalmente φ_g^s captura el requisito de que el robot siga desplazándose entre las regiones 2 y 16 a menos que encuentre un obstáculo.

Teniendo un escenario completamente detallado utilizando una fórmula LTL φ, la síntesis del problema consiste en la construcción de un autómata cuyos comportamientos satisfacen la fórmula. Está comprobado que este autómata es la doble exponencial del tamaño de esta fórmula [66]. Sin embargo, si se restringe el problema a la clase especial de fórmulas LTL se puede utilizar el algoritmo introducido en [55] el cual es de tiempo polinomial $O(n^3)$, donde n es el tamaño del espacio de estados. En este caso cada uno de los estados corresponde a una asignación de verdad admisible para el conjunto de proposiciones del entorno y del robot.

El proceso de síntesis es visto como un juego entre el robot y el entorno como el adversario. Partiendo de algún estado inicial, el robot y el entorno toman decisiones que determinan el estado siguiente de todo el sistema. La condición para ganar el juego está dada por una clase de fórmula ϕ de reactividad generalizada GR(1), que son fórmulas con estructura $(\Box\diamond p_1 \wedge \ldots \wedge \Box\diamond p_m) \rightarrow (\Box\diamond q_1 \wedge \ldots \wedge \Box\diamond q_n)$ donde p_i y q_i son una combinación Booleana de proposiciones atómicas. La manera de jugar es que en cada paso, primero el entorno hace una transición de

acuerdo a sus relaciones de transición y luego el robot hace su propia transición, si el robot logra satisfacer la fórmula ϕ sin importar lo que haga el ambiente, entonces el robot es el ganador y se puede obtener un autómata. Si el entorno logra hacer que el robot no satisfaga la fórmula ϕ, es decir, que después de los movimientos del entorno y del robot ϕ no sea cierto, entonces se puede decir que el entorno ganó y que el comportamiento deseado para el robot no se puede realizar o no es alcanzable. Para el caso se aborda, los estados iniciales de los jugadores están dados por φ_i^e y φ_i^s, las posibles transiciones que los jugadores pueden hacer están dadas por φ_t^e y φ_t^s y la condición para ganar está dada por la fórmula de ractividad generalizda (GR(1)) $\phi = (\varphi_g^e \Rightarrow \varphi_g^s)$. Según la fórmula que especifica las condiciones, el sistema puede ganar si φ_g^s es verdad o si φ_g^e es falso. En el primer caso, el comportamiento deseado para el robot se puede alcanzar y el sistema sería el ganador, pero el segundo caso implica que el entorno no logró alcanzar sus propios objetivos lo cual podría darse debido a que el robot se lo impidiera y en este caso no se podría garantizar el comportamiento del sistema. Además se tiene la opción de que el entorno no juegue limpio y el autómata generado ya no sea válido, lo que podría presentarse si el entorno ubicara un obstáculo en la región 2 o en la región 15.

El algoritmo de síntesis presentado en [55] toma la fórmula φ y primero verifica si esta es realizable; si es así, el algoritmo extrae un posible autómata que implementa una estrategia que el robot podría seguir para satisfacer la tarea deseada. El autómata generado por el algoritmo de síntesis es modelado como una tupla igual al explicado en la sección 5.5 en el cual $y_i = \gamma(q_i)$ es la etiqueta del iésimo estado en la ejecución y esta secuencia de etiquetas se utiliza para construir el camino discreto que debe seguir el robot y para activar o desactivar las diferentes acciones del mismo como encender o apagar la luz. El autómata obtenido en este caso al utilizar el algoritmo de síntesis es un autómata no determinista que se enfoca en alcanzar los objetivos en el menor número de transiciones.

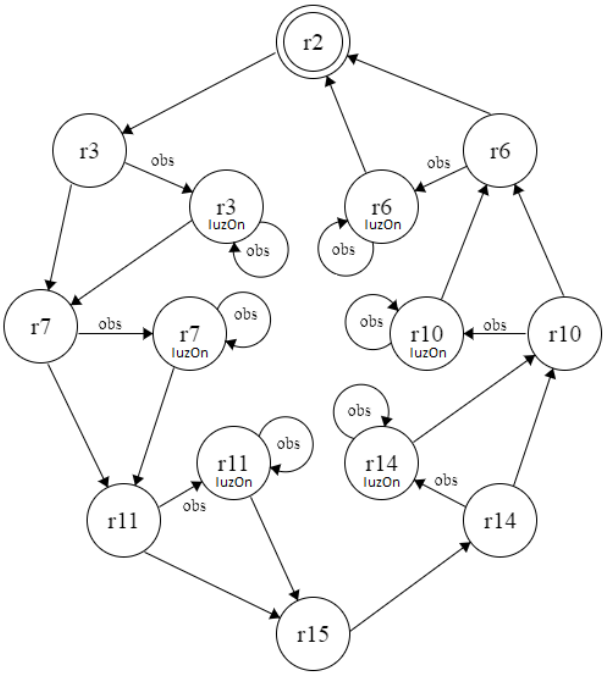

Figura 10. Autómata sintetizado para el caso 2.

En la figura 10 se muestra uno de los múltiples autómatas que se pueden obtener en la sintetización y que pueden cumplir con el comportamiento deseado, en donde los círculos representan los estados del autómata y las proposiciones que están escritas dentro de cada uno de los círculos son las etiquetas de los estados que indican las proposiciones de salida que son ciertas en ese estado. El estado inicial r_2 está denotado por un círculo doble y las flechas están etiquetadas con las proposiciones de sensado que deben ser verdaderas para que se lleve a cabo la transición. Las flechas sin etiqueta corresponden a la proposición (S^{obs}). Para este caso, el autómata hace que el robot se quede en la región en la que está si detecta un obstáculo, de lo contrario hace que avance a la región siguiente en la ruta. En caso de que el entorno se comportara de manera diferente a la supuesta, por ejemplo que el robot sense un obstáculo en r_{15}, el autómata no tendrá una transición definida y ya no será válido. El autómata obtenido no es el único que podría cumplir con las especificaciones utilizando el menor número de transiciones posibles [67].

En el algoritmo 2, la trayectoria que debe enviarse como variable de entrada puede ser calculada

Algorithm 2 Síntesis continua de políticas de control para la sección 9.2

Require: trayectoria, estadoActual, sensores()
Ensure: estadoSiguiente
1: sensado ← sensores()
2: estadoSiguiente ← GetEstadoSiguiente(trayectoria, estadoActual, sensado)
3: **return** estadoSiguiente

previamente utilizando el algoritmo 1. La función que calcula el estado siguiente requiere como argumento de entrada conocer el estado presente de la trayectoria en el que se encuentra el robot y además la información de sensado para así ejecutar las acciones definidas en caso de encontrar obstáculos (permanecer en el mismo punto hasta que el obstáculo desaparezca, re-calcular la ruta evadiendo el obstáculo, generar una alerta).

9.3. Múltiples robots y obstáculos fijos

Para el caso de un equipo de robots R idénticos modelados bajo los parámetros de la sección 8.2 que se mueven en un entorno rectangular. El entorno se divide en un conjunto finito de células disyuntas como el mostrado en la figura 5, y se ubican 2 robots representados por los círculos verdes, encargados de cumplir de manera conjunta la tarea dada en lenguaje natural como "Eventualmente visite la región a_2 pero no sin antes visitar la región a_3".

Inicialmente puede suponerse que con una adecuación al algoritmo 1 se puede resolver el problema, obtienendo el algoritmo 3.

Anque el algoritmo planteado en 3 se nota realizable, la necesidad de sincronización entre los autómatas de cada uno de los robots hace que el autómata del entorno crezca de manera $|P|^n * |B|$, siendo $|P|$ la cantidad de particiones presentes en el espacio de trabajo, n la cantidad de robots y $|B|$ la cantidad de estados presentes en el autómata de Büchi. Esto implica que al aumentar la cantidad de robots, aumenta el consumo de recursos para procesamiento hasta llegar un punto en el que se presenta una explosión combinacional de estados.

Otra solución a este caso implica elegir una secuencia de observaciones que satisfaga la fórmula LTL y luego generar la secuencia de activaciones adecuada en la RMPN que producen esa secuencia. Esta solución se basa en tres pasos principales:

Algorithm 3 Síntesis continua de políticas de control para la sección 9.3

Require: Entorno, Regionesinteres, especificacionLTL, robots[n]
Ensure: trayectoria

```
P ← GetPartTriang (Entorno, Regionesinteres)
AutRobot(1) ← GetAutEnt(P, robot(1))
...
AutRobot(n) ← GetAutEnt(P, robot(n))
AutEntorno ← GetAutEnt(AutRobot(1),...,AutRobot(n))
if AutEntorno = false then
   return Error al definir el entorno
end if
AutBuchi ← GetAutBuchi(especificacionLTL)
if AutBuchi = false then
   return Error en formula LTL
end if
AutGeneral ← GetAutGeneral(AutEntorno, AutBuchi)
trayectoria ← findRunAcep(AutGeneral)
return trayectoria
```

1. Se elige un buen prefijo finito r (llamado run) del autómata Büchi correspondiente a la fórmula LTL.

2. Para cada transición de la ejecución r, se busca una secuencia de activaciones para el modelo RMPN de modo que las observaciones generadas produzcan la transición elegida.

3. Las estrategias de movimiento de los robots se obtienen al concatenar las secuencias de disparos del paso 2 e imponer momentos de sincronización entre estas secuencias.

Paso 1: Encontrar un conjunto de rutas de B, por ejemplo, utilizando un algoritmo de ruta k-shortest [68] en el gráfico de adyacencia correspondiente a las transiciones de B. Luego, los pasos 2 y 3 pueden iterarse, tomando en cada iteración otra ejecución r del conjunto construido de posibles ejecuciones de B. Una vez que se puede seguir una ejecución debido a las observaciones de RMPN (el paso 2 tuvo éxito), la iteración de los pasos 1 - 3 puede detenerse y la solución devuelta está dada por la ejecución actualmente elegida de B, es decir, por $r = s_0 s_1 ... s_{L_r}$, donde L_r es la longitud de r.

Paso 2: para habilitar la transición $s_j \rightarrow s_{j+1} \text{en } B, j = 0, ..., L_r - 1$, las siguientes dos condiciones deberían ser válidas: (1) El sistema RMPN debe alcanzar una marca final m que genera

cualquier observación del conjunto $\rho(s_j, s_{j+1}) \subseteq 2^{\Pi}$; (2) Las marcas RMPN intermedias deberían generar solo observaciones en el conjunto $\rho(s_j, s_j) \subseteq 2^{\Pi}$, de modo que s_j no se deje en otros estados que no sean s_{j+1}. Para verificar una observación en una marca m alcanzable dada, se define para cada observación $\pi_i \in \Pi$ una variable binaria x_i tal que:

$$x_i = \begin{cases} 1, & if\ v_i \cdot m > 0 \\ 0, & si\ otra\ cosa \end{cases} \tag{14}$$

Las siguientes 2 condiciones asignan el valor correcto a x_i:

$$\begin{cases} N \cdot x_i & \geq v_i \cdot m \\ x_i & \leq v_i \cdot m \end{cases} \tag{15}$$

Donde N es un número muy grande. Se resalta que si $vi \cdot m > 0$, la primera restricción de la ecuación 15 es $x_i = 1$, mientras que la segunda restricción de la ecuación 15 asegura que $x_i = 0$ si $v_i \cdot m = 0$.

Para derivar descripciones formales apropiadas equivalentes a las condiciones expresadas en las ecuaciones 14 y 15, se considera un subconjunto genérico $S \subseteq 2^{\Pi}$. Se desea que el conjunto de observaciones activas en una marca m se incluya en S. El conjunto S se puede ver como una disyunción de conjunciones de proposiciones de Π para convertirlo en una Forma Normal de Conjunciones (CNF) por doble negación, es decir las observaciones en m no deberían pertenecer a $2\Pi \backslash S$. Teniendo ya el conjunto CNF, se puede utilizar [34] para escribir un conjunto de desigualdades que deban sostenerse simultáneamente de modo que las observaciones en m pertenezcan a S.

En el algoritmo 4 se utiliza el software CPLEX proveído por IBM para resolver el problema MILP que se plantea.

Algorithm 4 Síntesis continua de políticas de control para la sección 9.3

Require: Entorno, Regionesinteres, especificacionLTL, CPLEX
Ensure: secuencia[secuencia de activaciones para cada robot]

```
P ← GetPartTriang (Entorno, Regionesinteres)
Q ← GetModeloRMPN(P)
if Q = false then
    return Error al definir Q
end if
AutBuchi ← GetAutBuchi(especificacionLTL)
if AutBuchi = false then
    return Error en formula LTL
end if
trayectoria ← findRunAcep(AutBuchi)
while trayectoria ≠ ∅ do
    construirπ()
    construirΠ()
    X ← GetXvector(π, Π)
    for j = 0, 1, ..., L_r − 1 do
        MILP ← formularMILP(X)
        resolverMILP(CPLEX)
        if σ es aplicable then
            actualizar(secuencia[])
        end if
    end for
    if se visitan todos los estados de trayectoria then
        return secuencia
    else
        trayectoria ← otra(trayectoria)
    end if
end while
```

Parte IV

MARCO EXPERIMENTAL

El procedimiento para aplicar el experimento consiste en tres pasos fundamentales recopilados en el algoritmo 5 los cuales son: a) Selección del algoritmo a ejecutar dependiendo del problema abordado, b) Ejecución del algoritmo y c) Entrega de resultados. Se configura el experimento en la implementación de los algoritmos propuestos en las secciones 9.1, 9.2, 9.3 y 5 utilizando el software de programación Matlab instalado sobre un sistema operativo Windows 7 en una computadora portátil de referencia ASUS X555L con procesador Intel Core i7-4510U, up 3.1 GHz y memoria RAM de 8GB, para lo cual se utilizó como apoyo un Toolbox para movimiento de robots [69].

El algoritmo 5 requiere para su ejecución una valoración inicial del problema de planificación de movimiento en la que se debe incluir especialmente la fórmula LTL extraida en la línea 2, en la que describe el entorno, las opciones de movimiento del sistema y la especificación que se desea alcanzar, y también la cantidad de robots con los que se cuenta para alcanzar dicha tarea (información extraida en la línea 1). En las siguientes líneas del algoritmo (5-26) se evalúa la cantidad de robots, el tipo de tarea que se le solicita cumplir al sistema y las preferencias de ejecución del usuario, para así seleccionar el algoritmo adecuado para la solución del problema.

10. EXPERIMENTOS

En esta sección se plantean un total de 20 simulaciones enmarcadas en 9 casos particulares y un ejemplo de planificación de rutas en una aplicación real. La estructura de las simulaciones presentadas puede observarse en la tabla 2.

Algorithm 5 Algoritmo general para síntesis continua de políticas de control

Require: ProblemPlaniMovimiento[], parametros[]
Ensure: ConjuntodeTrayectoriasSolucionalProblema()

```
Robots ← ProblemPlaniMovimiento[numeroRobots]
LTL ← ProblemPlaniMovimiento[tareaLTL]
Pref ← preferencias
tipoTarea ← SintesisLTL(LTL)
if Robots < 2 then
    if tipoTarea = RG(1) then
        resultado ← ejecutar(algoritmo2)
    else
        if Pref = alg4 then
            resultado ← ejecutar(algoritmo4)
        else
            resultado ← ejecutar(algoritmo1)
        end if
    end if
    return resultado
end if
if Robots >= 2 then
    if Pref = alg3 then
        resultado ← ejecutar(algoritmo3)
    else
        resultado ← ejecutar(algoritmo4)
    end if
    return resultado
end if
ConjuntodeTrayectoriasSolucionalProblema() ← resultado
return ConjuntodeTrayectoriasSolucionalProblema()
```

# Simulación	Sección	Rutas calculadas
1	10.1	Figura 12
2	10.2	Figura 14
3	10.3	Figura 15
4	10.3.1	Figura 16
5	10.3.2	Figura 17
6	10.3.3	Figura 18(Superior izquierda)
7		Figura 18(Superior derecha)
8		Figura 18(Inferior izquierda)
9		Figura 18(Inferior derecha)
10	10.4	Figura 19
11	10.5	Figura 21
12	10.6	Figura 22
13	10.7	Figura 23
14	10.8	Figura 24
15	10.8.1	Figura 26(Superior izquierda)
16		Figura 26(Superior derecha)
17		Figura 26(Inferior izquierda)
18		Figura 26(Inferior derecha)
19	10.9	Figura 27
20	10.10	Figura 29

10.1. Caso1

Para el problema definido en la sección 8.1 se plantea una solución que consiste en generar inicialmente un entorno de trabajo conocido con un único robot, como se puede observar en la figura 11.

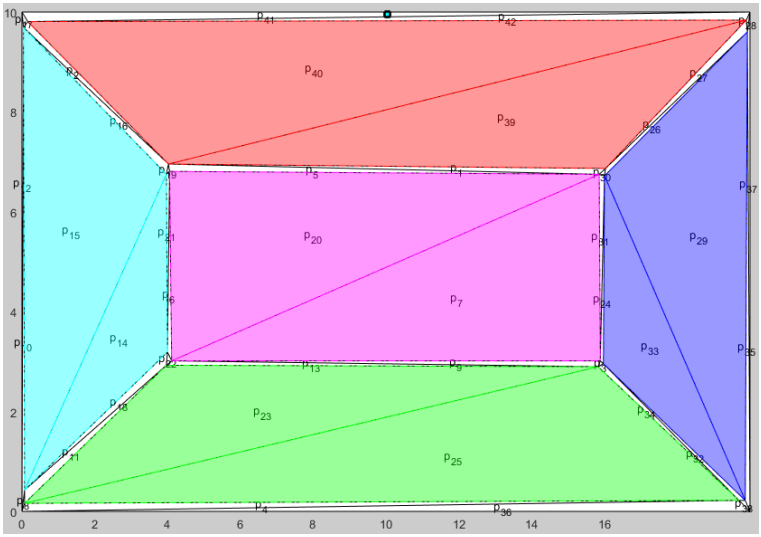

Figura 11. Espacio de trabajo.

El problema planteado inicialmente en la sección 8.1 con la especificación LTL mostrada en la ecuación 16, puede resolverse rápidamente utilizando el algoritmo 1 implementado en Matlab, con lo cual se obtiene como resultado la ruta trazada en la figura 12 y los datos depositados en la tabla 3.

$$\varphi = \Diamond a_1 \wedge \Diamond a_2 \wedge \Diamond a_3 \wedge \Diamond a_5 \wedge \Box!(a_4) \tag{16}$$

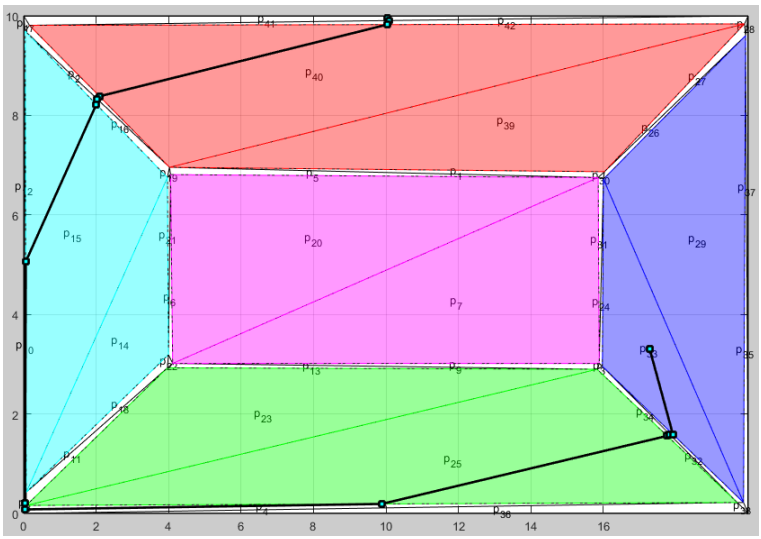

Figura 12. Trayectoria obtenida con el algoritmo 1 para la fórmula 16

Criterio	Resultado
Sistema de transiciones para un único robot	42 estados
Tiempo utilizado para generar el diagrama	0.0321067 segundos
Autómata de Büchi solución a la fórmula LTL	16 estados
Autómata resultante para el sistema completo	672 estados
Tiempo utilizado para generar el autómata del sistema completo	0.0821076 segundos
Tiempo utilizado para encontrar una trayectoria aceptable en el autómata	0.194141 segundos

Tabla 3. Caso 1, algoritmo 1, fórmula 16

10.2. Caso 2

Para el problema planteado en la sección 9.2, se plantea un entorno de trabajo particionado como el que se muestra en la figura 13, compuesto por 16 regiones de interés y se plantea como especificación, la necesidad de desplazamiento del robot entre las regiones 2 (región de color azul en la fila superior) y 15 (región de color verde en la última fila). La especificación de desplazamiento está dada por la fórmula LTL 17.

$$\varphi = \Diamond a_2 \wedge \Diamond a_{15} \tag{17}$$

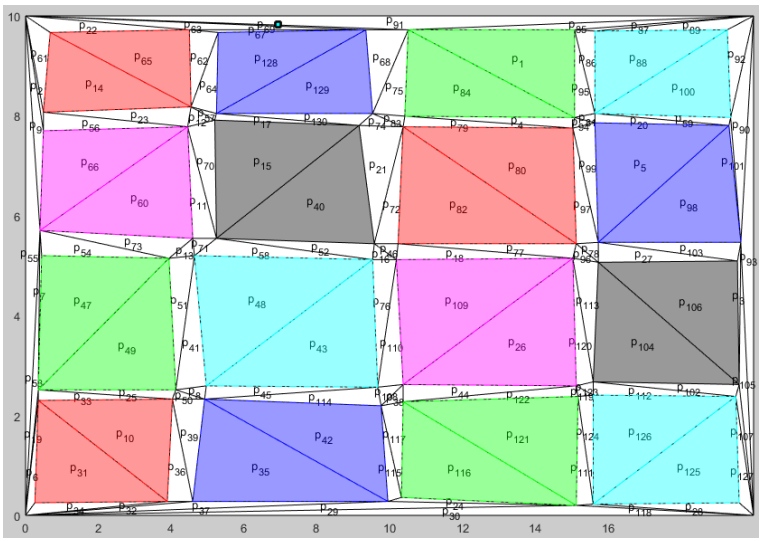

Figura 13. Entorno de trabajo particionado para la sección 9.2

La trayectoria obtenida al ejecutar el algoritmo 2 para resolver este problema, puede apreciarse en la figura 14 y los respectivos datos asociados a la ejecución en la tabla 4.

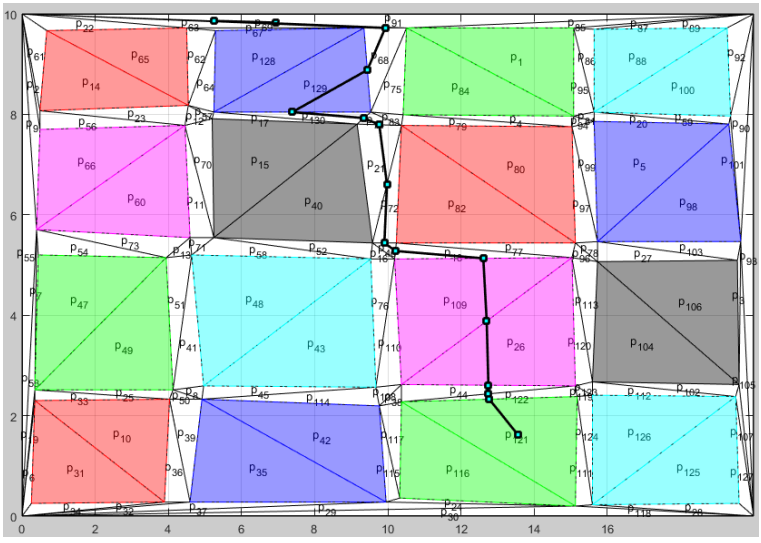

Figura 14. Trayectoria obtenida con el algoritmo 1 para la fórmula 17

Criterio	Resultado
Sistema de transiciones para un único robot	130 estados
Tiempo utilizado para generar el diagrama	2.32755 segundos
Autómata de Buchi solución a la fórmula LTL	4 estados
Autómata resultante para el sistema completo	520 estados
Tiempo utilizado para generar el autómata del sistema completo	0.157414 segundos
Tiempo utilizado para encontrar una trayectoria aceptable en el autómata	0.39446 segundos

Tabla 4. Resultados caso 2, algoritmo 1, fórmula 17

10.3. Caso 3

Se recurre entonces a solucionar el problema a través de la ejecución del algoritmo 2 y se incluye en la especificación una condición de evasión de obstáculos como se puede apreciar en la fórmula18, obteniendo así un nuevo conjunto de resultados conformado por la figura 15 y la tabla 5 en los que se evidencia el cumplimiento de las especificaciones.

$$\varphi = \Diamond a_2 \wedge \Diamond a_{15} \wedge \Box!(a_1 \vee a_3 \vee a_4 \vee ... \vee a_{14} \vee a_{16}) \tag{18}$$

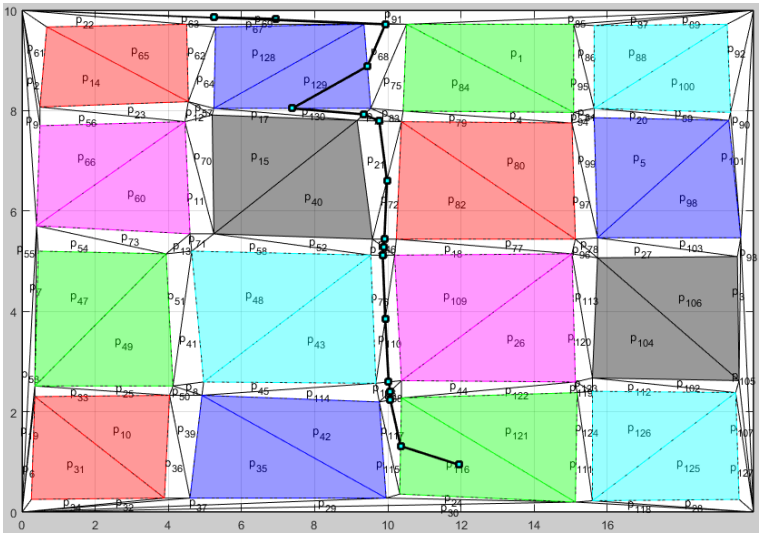

Figura 15. Trayectoria obtenida con el algoritmo 2 para la fórmula 18

Criterio	Resultado
Sistema de transiciones para un único robot	130 estados
Tiempo utilizado para generar el diagrama	0.247121 segundos
Autómata de Büchi solución a la fórmula LTL	4 estados
Autómata resultante para el sistema completo	520 estados
Tiempo utilizado para generar el autómata del sistema completo	0.0619943 segundos
Tiempo utilizado para encontrar una trayectoria aceptable en el autómata	0.155664 segundos

Tabla 5. Resultados caso 3, algoritmo 2, fórmula 18

10.3.1. Variación 1 del caso 3

La figura 16 y la tabla 6 contienen la trayectoria y los datos resultantes para la fórmula LTL 19 en la que se incluye la región 12 como una región de interés. Sin embargo, el orden en que se alcanzan las regiones no está especificado.

$$\varphi = \Diamond a_2 \wedge \Diamond a_{15} \wedge \Diamond a_{12} \wedge \Box!(a_1 \vee a_3 \vee a_4 \vee ... \vee a_{11} \vee a_{13} \vee a_{14} \vee a_{16}) \tag{19}$$

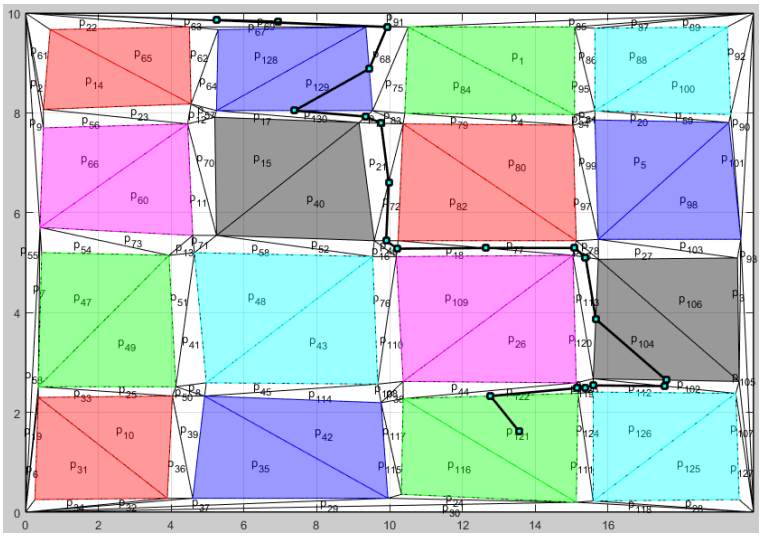

Figura 16. Trayectoria obtenida con el algoritmo 2 para la fórmula 19

Criterio	Resultado
Sistema de transiciones para un único robot	130 estados
Tiempo utilizado para generar el diagrama	0.201292 segundos
Autómata de Buchi solución a la fórmula LTL	8 estados
Autómata resultante para el sistema completo	1040 estados
Tiempo utilizado para generar el autómata del sistema completo	0.166681 segundos
Tiempo utilizado para encontrar una trayectoria aceptable en el autómata	0.389911 segundos

Tabla 6. Resultados de la variación 1 del caso 3

10.3.2. Variación 2 del caso 3

En la figura 17 y en la tabla 7 puede apreciarse la trayectoria y los datos correspondientes a la solución hallada al sintetizar la ecuación 20, a través del algoritmo 2 en Matlab.

$$\varphi = \Diamond(a_2 \wedge \Diamond(a_{15} \wedge \Diamond(a_{12} \wedge (!a_{12})Ua_{15}))) \wedge \Box!(a_1 \vee a_3 \vee a_4 \vee ... \vee a_{11} \vee a_{13} \vee a_{14} \vee a_{16}) \quad (20)$$

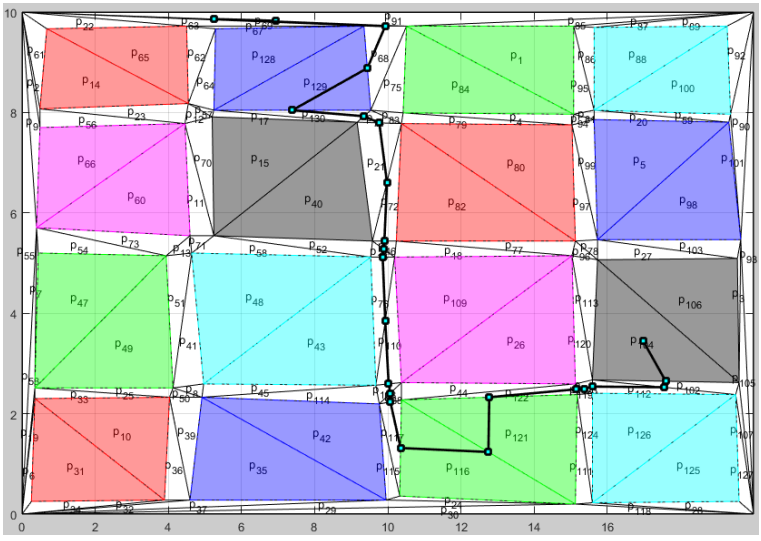

Figura 17. Trayectoria obtenida con el algoritmo 2 para la fórmula 20

Criterio	Resultado
Sistema de transiciones para un único robot	130 estados
Tiempo utilizado para generar el diagrama	0.145803 segundos
Autómata de Büchi solución a la fórmula LTL	6 estados
Autómata resultante para el sistema completo	780 estados
Tiempo utilizado para generar el autómata del sistema completo	0.114047 segundos
Tiempo utilizado para encontrar una trayectoria aceptable en el autómata	0.334383 segundos

Tabla 7. Resultados de la variación 2 del caso 3

10.3.3. Pruebas caso 3

En el bloque de figuras 18 se valida la capacidad del sistema para encontrar una ruta de manera automática para cuatro configuraciones de obstáculos diferentes en un mismo entorno.

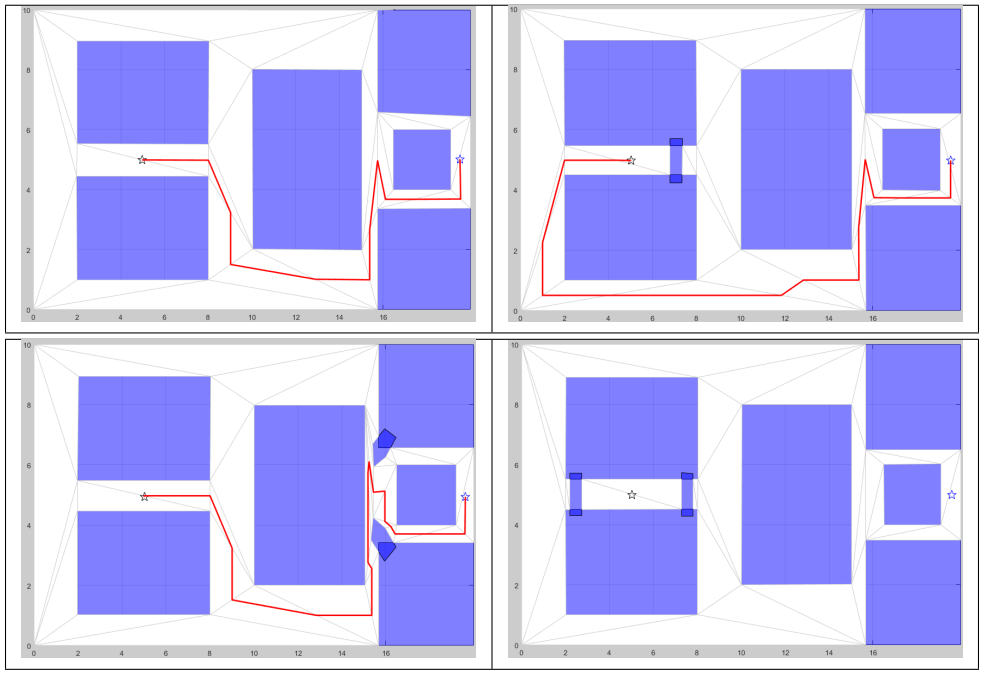

Figura 18. Soluciones del algoritmo 2 variando la posición de los obstáculos.

10.4. Caso 4

En la figura 19 y en la tabla 8 puede apreciarse la trayectoria y los datos respectivamente, asociados a la solución del problema 8.1 utilizando el algoritmo 4 desarrollado en la sección

9.3. En este caso, la fórmula LTL podría expresarse en una forma FNC como se muestra en la ecuación 21, en donde las proposiciones $!D$ y $!d$ indican la restricción que tiene el robot para visitar o terminar en la región a_4 durante la ejecución.

$$A \wedge B \wedge C \wedge E \wedge !D \wedge !d \tag{21}$$

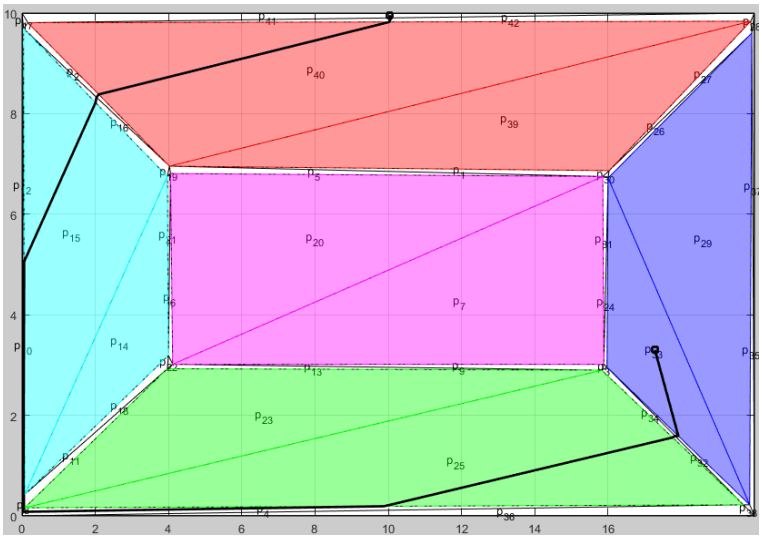

Figura 19. Trayectoria obtenida con el algoritmo 4 para la fórmula 21

Criterio	Resultado
Cantidad de lugares de la red de Petri	42 lugares
Cantidad de transiciones de la red de Petri	122 transiciones
Tiempo utilizado para generar la red de Petri	0.508616 segundos
Variables presentes en el problema MILP	2635 variables
Restricciones de igualdad	672 igualdades
Restricciones de desigualdad	740 desigualdades
Tiempo utilizado para crear el problema MILP	0.508696 segundos
Tiempo utilizado para resolver el problema MILP	1.27768 segundos
Solución Robot 1	41,42,40,2,16,15,10,8,4,25,34,32,33

Tabla 8. Resultados caso 4, algoritmo 4, fórmula 21

10.5. Caso 5

El entorno de trabajo de la figura 20 consta de 3 regiones de interés y 2 robots que deben cumplir una tarea expresada como una fórmula LTL de manera conjunta en la menor cantidad de transiciones para el caso de un sistema de estados y transiciones y análogamente con la menor cantidad de activaciones en una red de Petri.

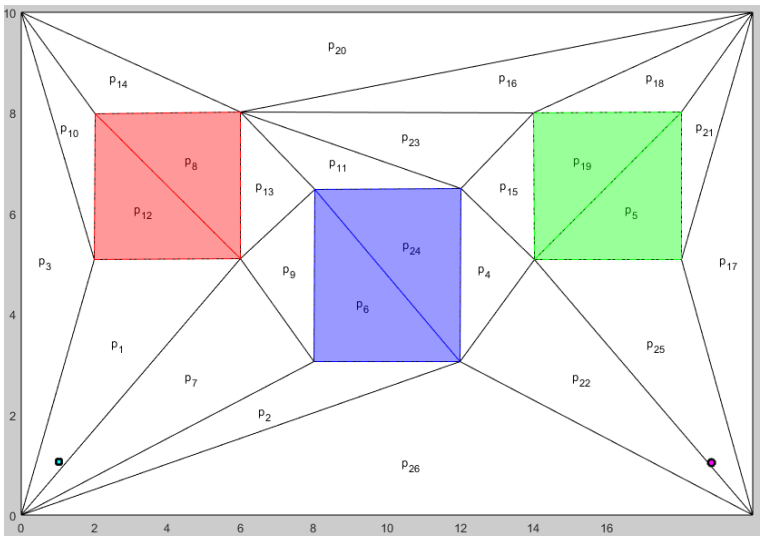

Figura 20. Entorno de trabajo particionado con múltiples obstáculos y dos robots.

En la figura 21 y en la tabla 9 pueden apreciarse la trayectoria y los datos resultantes respectivamente de resolver la fórmula LTL 22 a través del algoritmo 3 para el entorno mostrado en la figura 20 con 2 robots.

$$\varphi = \Diamond P_1 \wedge \Diamond P_2 \wedge \Diamond P_3 \tag{22}$$

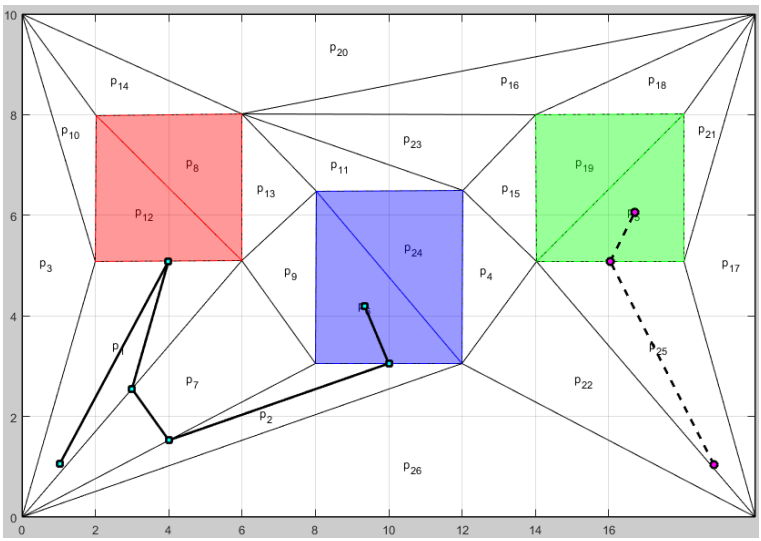

Figura 21. Trayectorias obtenidas con el algoritmo 3 para la fórmula 22

Criterio	Resultado
Sistema de transiciones para un único robot	26 estados
Sistema de transiciones para el equipo de robots	676
Tiempo utilizado para generar el diagrama	0.800783 segundos
Autómata de Büchi solución a la fórmula LTL	8 estados
Autómata resultante para el sistema completo	5408 estados
Tiempo utilizado para generar el autómata del sistema completo	5.61371 segundos
Tiempo utilizado para encontrar una trayectoria aceptable en el autómata	11.232 segundos

Tabla 9. Resultados caso 5, algoritmo 3, fórmula 22

10.6. Caso 6

En la figura 22 y en la tabla 10 pueden apreciarse la trayectoria y los datos resultantes respectivamente de resolver la fórmula LTL en forma FNC 23 a través del algoritmo 4 para el entorno mostrado en la figura 20 con 2 robots.

$$\varphi = A \wedge B \wedge C \tag{23}$$

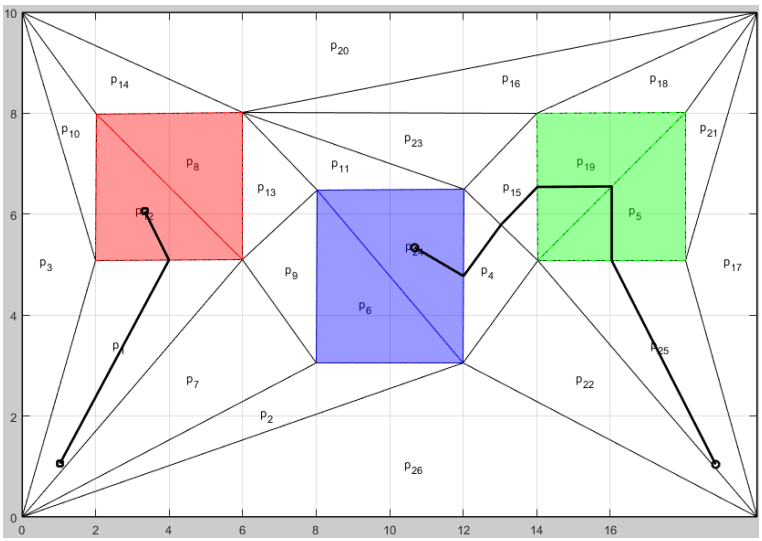

Figura 22. Trayectorias obtenidas con el algoritmo 4 para la fórmula 23

Criterio	Resultado
Cantidad de lugares de la red de Petri	26 lugares
Cantidad de transiciones de la red de Petri	74 transiciones
Tiempo utilizado para generar la red de Petri	0.244573 segundos
Variables presentes en el problema MILP	1607 variables
Restricciones de igualdad	416 igualdades
Restricciones de desigualdad	457 desigualdades
Tiempo utilizado para crear el problema MILP	0.244648 segundos
Tiempo utilizado para resolver el problema MILP	1.16772 segundos
Solución Robot 1	1,12
Solución Robot 2	25, 5, 19, 15, 4, 24

Tabla 10. Resultados caso 6, algoritmo 4, fórmula 23

10.7. Caso 7

Síntesis de la fórmula LTL 22 en el entorno de la figura 23 con las mismas 3 regiones de interés, pero ahora con 3 robots.

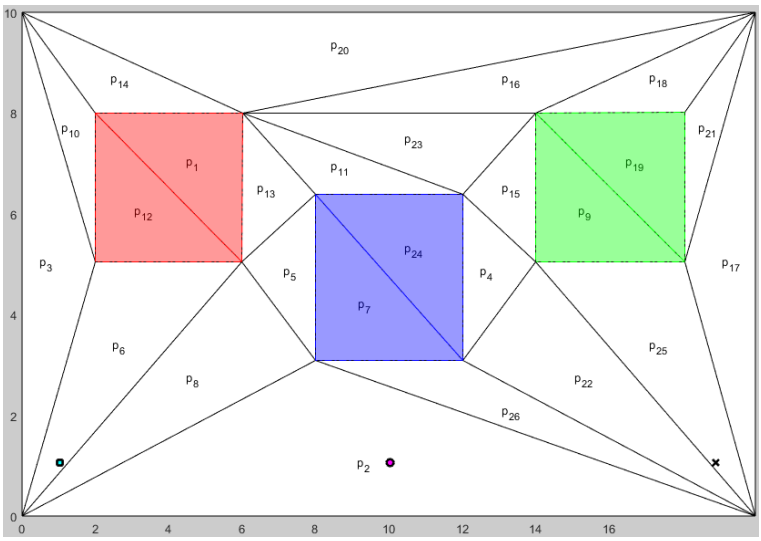

Figura 23. Entorno de trabajo particionado con múltiples obstáculos y tres robots

10.8. Caso 8

Al sintetizar a través del algoritmo 4 la fórmula LTL 23, en el entorno de la figura 23 con las mismas 3 regiones de interés, pero ahora con 3 robots, se obtiene como resultado las trayectorias que se observan en la figura 24 y los datos depositados en la tabla 11.

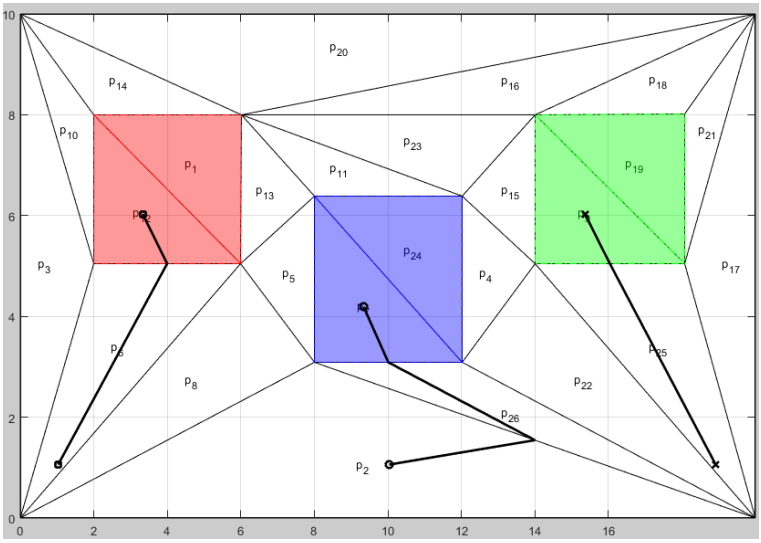

Figura 24. Trayectorias obtenidas con el algoritmo 4 para la fórmula 23

Criterio	Resultado
Cantidad de lugares de la red de Petri	26 lugares
Cantidad de transiciones de la red de Petri	74 transiciones
Tiempo utilizado para generar la red de Petri	0.284 segundos
Variables presentes en el problema MILP	1607 variables
Restricciones de igualdad	416 igualdades
Restricciones de desigualdad	457 desigualdades
Tiempo utilizado para crear el problema MILP	0.284084 segundos
Tiempo utilizado para resolver el problema MILP	0.2577 segundos
Solución Robot 1	6, 12
Solución Robot 2	2, 26, 7
Solución Robot 3	25, 9

Tabla 11. Resultados caso 8 con tres robots, algoritmo 4, fórmula 23

10.8.1. Pruebas para el caso 8

La fórmula 24, entregada en forma FNC, expresa que el equipo de robots debe alcanzar de manera conjunta las regiones A, B, C, D correspondientes con las regiones de colores rojo, violeta, verde y azul en la figura 25.

$$\varphi = A \wedge B \wedge C \wedge D \tag{24}$$

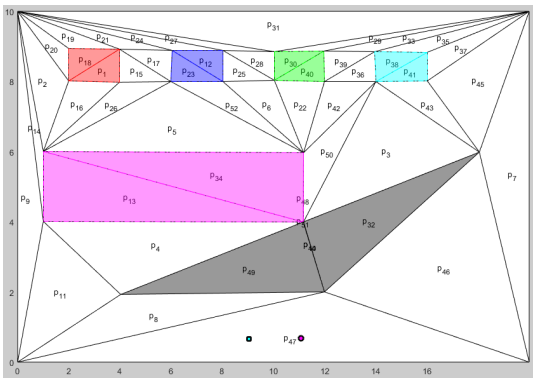

Figura 25. Entorno para pruebas con 2, 3, 4 y 5 robots

Al ejecutar el algoritmo 4 para dar solución a este problema con 2, 3, 4 y 5 robots, se obtienen las trayectorias que se muestran en la figura 26.

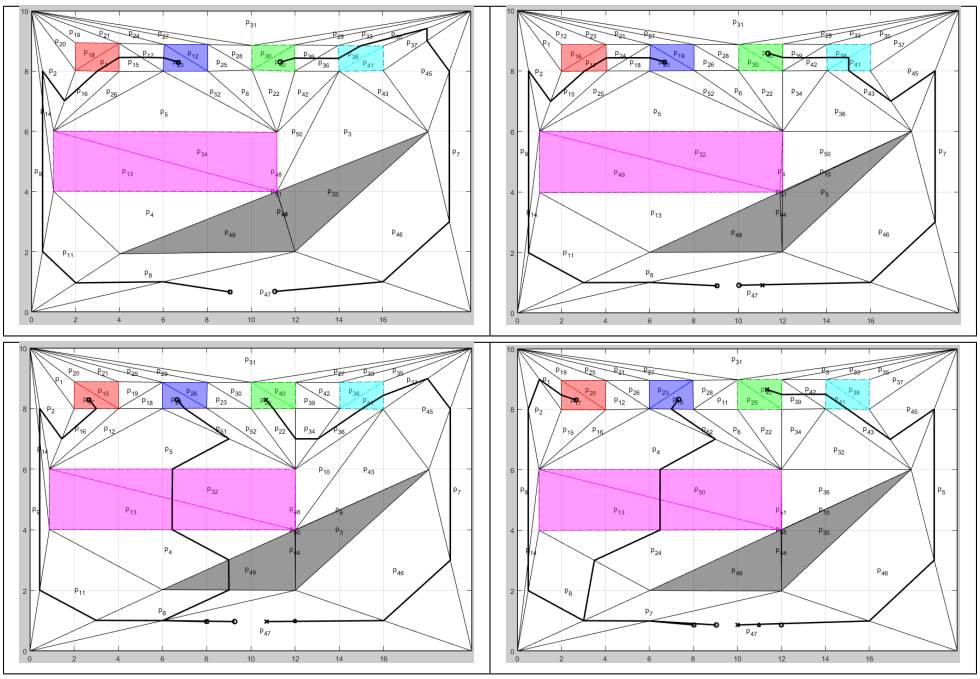

Figura 26. Trayectorias calculadas con el algoritmo 4, entorno 25 y fórmula 24.

En la tabla 12 puede apreciarse la recopilación de los datos obtenidos para cada una de las ejecuciones del algoritmo 4 en el cálculo de las trayectorias depositadas en la figura 26. En este caso, las filas representan representan los criterios de medición, en donde P representa el tamaño de lugares que contiene la RdP resultante, T representa la cantidad de transiciones de la RdP, t_p representa el tiempo que se tarda el sistema para construir la RdP, V_{MILP} representa la cantidad de variables que se generan para el problema MILP, $=$ representa la cantidad de restricciones de igualdad, $\neq$ representa la cantidad de restricciones de desigualdad, $t_{c(MILP)}$ representa el tiempo necesario para construir el problema MILP, $T_{r(MILP)}$ representa el tiempo necesario para resolver el problema MILP, T_{R_1}, T_{R_2}, T_{R_3}, T_{R_4} y T_{R_5} representan los conjuntos de activaciones correspondientes a las trayectorias de cada robot. Las columnas representan los 4 casos planteados, es decir, para 2, 3, 4 y 5 robots.

Criterio	2 robots	3 robots	4 robots	5 robots
P	52	52	52	52
T	152	152	152	152
t_p	0.286772	0.306158	0.296011	0.298367
V_{MILP}	2053	2053	2053	2053
$=$	520	520	520	520
$\neq$	600	600	600	600
$t_{c(MILP)}$	0.28688	0.30624	0.296094	0.298453
$t_{r(MILP)}$	0.264835	0.184947	0.251981	0.248381
T_{R_1}	47,8,11,9,14,2,-16,1,15,17,23	47,8,11,14,9,2,-15,17,24,18,20	47,8,11,9,14,2,-16,17	47,7,8,14,9,2,-1,17
T_{R_2}	47,46,7,45,37-,35,38,36,39,40	47,46,7,45,43,-41,38,42,39,40	47,8,49,4,13,-32,5,51,24	47,7,8,24,13,-50,4,52,18
T_{R_3}	– –	47	47,46,7,45,37,-41,36,34,22,28	47,46,5,45,43,-41,42,39,40
T_{R_4}	– –	– –	47	47
T_{R_5}	– –	– –	– –	47

Tabla 12. Bloque de resultados para las pruebas del caso 8

10.9. Caso 9

Para este caso se propone ejecutar la fórmula FNC 25 con 4 robots en el entorno de la figura 25.

$$\varphi = A \wedge B \wedge C \wedge D \wedge !E \wedge !F \wedge !a \wedge !b \wedge !c \wedge !d \tag{25}$$

La fórmula 25 especifica que el conjunto de robots debe alcanzar las regiones (A, B, C, D) sin pasar por las regiones (E, F) y además especifica que al finalizar la ejecución de las trayectorias, no puede haber ningún robot en las regiones (a, b, c, d).

En la figura 27 y en la tabla 13 puede apreciarse la trayectoria y los datos resultantes al aplicar el algoritmo 4 para la especificación dada en la fórmula FNC 25.

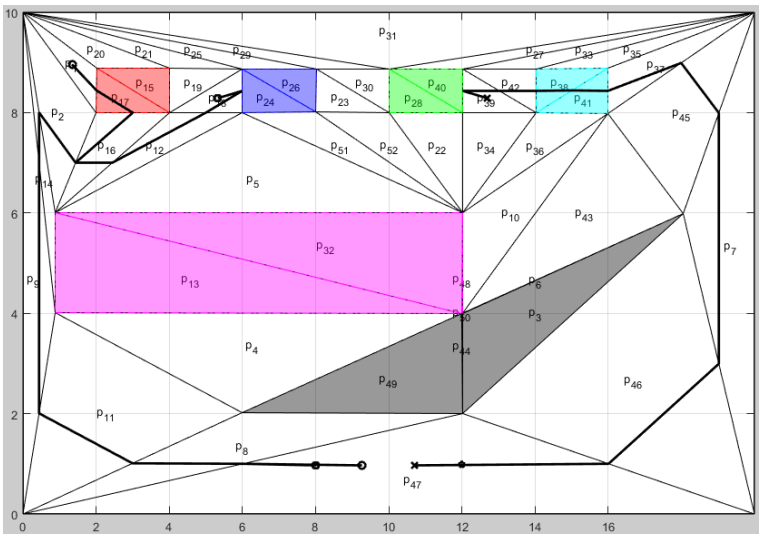

Figura 27. Trayectorias obtenidas con el algoritmo 4 para la fórmula 25

Criterio	Resultado
Cantidad de lugares de la red de Petri	52 lugares
Cantidad de transiciones de la red de Petri	152 transiciones
Tiempo utilizado para generar la red de Petri	0.312603 segundos
Variables presentes en el problema MILP	2053 variables
Restricciones de igualdad	520 igualdades
Restricciones de desigualdad	606 desigualdades
Tiempo utilizado para crear el problema MILP	0.312686 segundos
Tiempo utilizado para resolver el problema MILP	0.129306 segundos
Solución Robot 1	47,8,11,9,14,2,16,12,18,24,18
Solución Robot 2	47,8,11,9,14,2,16,17,1
Solución Robot 3	47,46,7,45,37,41,38,42,39,40,39
Solución Robot 4	47

Tabla 13. Resultados caso 9, algoritmo 4, fórmula 25

10.10. Ejemplo de aplicación

Suponga una empresa que presta los servicios de recolección de basuras en diferentes lugares de la ciudad. La empresa cuenta con 8 camiones para la recolección de la basura que se comienzan la ruta en la mañana desde una bodega ubicada en la ciudad y finalizan la ruta en el relleno sanitario ubicado en las afueras de la ciudad. Se desea calcular las trayectorias de cada uno de los camiones para garantizar que se cubren todas las áreas de interés en el menor tiempo

posible.

En este caso, los camiones se representan por los robots, la tarea global esta dada en lenguaje natural como “Recoger las basuras de todas las áreas de interés y depositarlas en el relleno sanitario en el menor tiempo posible”, la ciudad representaría el entorno y las trayectorias serían el resultado obtenido de la ejecución del algoritmo 4. En la figura 28 se plantea un entorno de ciudad con 12 regiones de interés y 8 robots (recolectores). En este caso, la bodega se encuentra en la parte inferior izquierda y el relleno sanitario en la parte superior derecha.

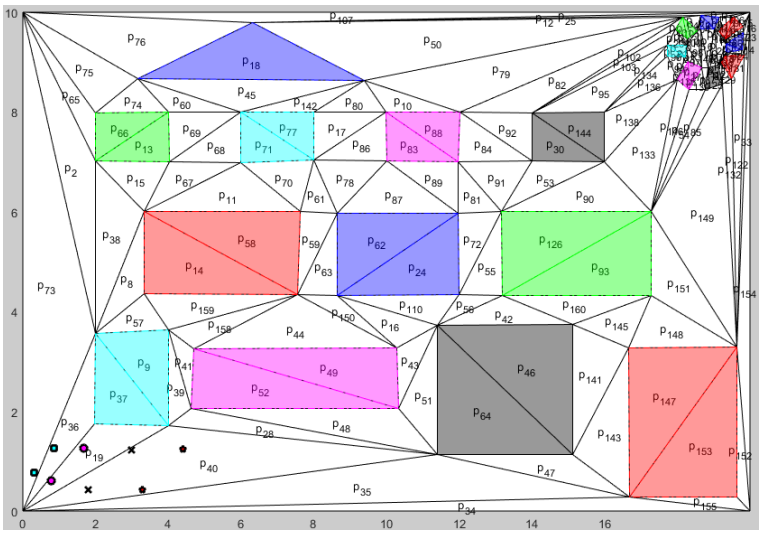

Figura 28. Entorno propuesto para el ejemplo planteado en la sección 10.10

Inicialmente se plantea la necesidad de entregar un paquete en cada una de las regiones de interés existentes en la parte superior. Esta tarea está expresada por la fórmula FNC 26.

$$A \wedge B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I \wedge J \wedge K \wedge L \wedge m \wedge n \wedge o \wedge p \wedge q \wedge r \wedge s \wedge t \qquad (26)$$

La ecuación 26 especifica que el equipo de robots debe visitar las regiones $(A, B, C, D, E, F, G-, H, I, J, K, L)$ y al finalizar las visitas, los robots deben terminar su recorrido a las regiones (m, n, o, p, q, r, s, t) que se encuentran en la parte superior derecha del entorno de la figura 28. La figura 29 muestra las posibles trayectorias que cumplen con la especificación y la tabla 14 muestra los datos obtenidos de la simulación.

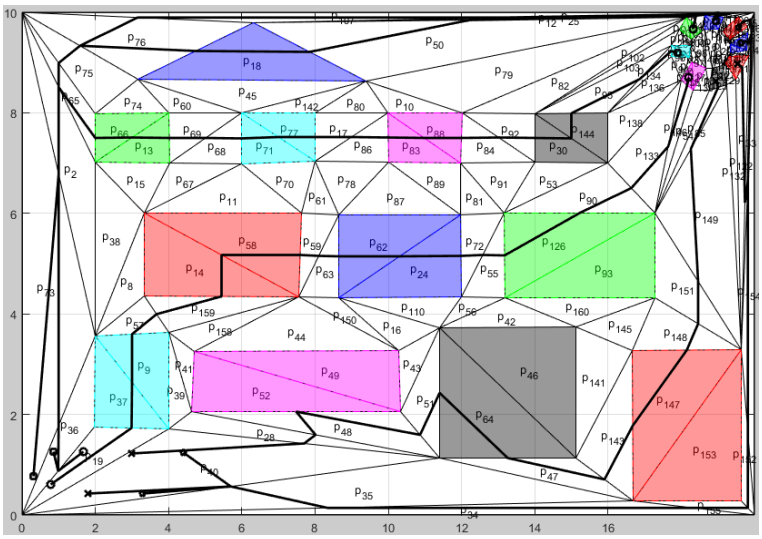

Figura 29. Trayectorias obtenidas con el algoritmo 4 para la fórmula 26

Criterio	Resultado
Cantidad de lugares de la red de Petri	160 lugares
Cantidad de transiciones de la red de Petri	476 transiciones
Tiempo utilizado para generar la red de Petri	28.4371 segundos
Variables presentes en el problema MILP	12761 variables
Restricciones de igualdad	3200 igualdades
Restricciones de desigualdad	3460 desigualdades
Tiempo utilizado para crear el problema MILP	28.4376 segundos
Tiempo utilizado para resolver el problema MILP	47.5864 segundos
Solución Robot 1	36,1 9,36,73,2,65,75,-76,107,25,113,109
Solución Robot 2	19,36,73,2,65,75,76,-18,50,12,4,105
Solución Robot 3	40,28,48,52,48,51,64,47,143,-147,148,151,149,85,125
Solución Robot 4	40,35,34,155,152,154,-33,123,116,115,156
Solución Robot 5	73,2,65,66,13,69,68,71,77,17,86,-83,88,84,92,30,144,95,134,96
Solución Robot 6	19,37,9,57,159,14,58,59,63,62,-24,72,55,126,90,133,146,135
Solución Robot 7	40,35,34,155,152,-154,33,122,131
Solución Robot 8	40,35,34,155,152,154,-33,123,114,121

Tabla 14. Tabla de resultados para el ejemplo de aplicación práctico

11. RESULTADOS

11.1. Caso 10.1

En la tabla de resultados 3 se evidencia que el sistema de transiciones para un solo robot se desarrolla en función de la cantidad de particiones que se generen en el espacio de trabajo. Además, el autómata resultante del sistema completo depende de la multiplicación de la cantidad de estados presentes en el autómata de Büchi solución de la fórmula LTL, que en este caso es 16 y la cantidad de estados del sistema de transiciones. Finalmente, se nota que el tiempo utilizado para encontrar una trayectoria aceptable en el autómata es uno de los factores que más recursos consume en el proceso y esto se debe, a que el algoritmo para garantizar una respuesta óptima, debe evaluar todas las rutas posibles entre el estado inicial y el estado final del autómata que representa al sistema completo.

11.2. Caso 10.2

En este caso, el mayor tiempo del proceso está asociado a la generación del diagrama de estados y transiciones para el robot y este diagrama está ligado directamente a la cantidad de regiones en las que se divide el espacio y a la relación de adyacencia entre las mismas. Además se nota que la trayectoria obtenida no evade los obstáculos presentes en el espacio de trabajo de la figura 13 correspondientes a las regiones 1, 3,..., 14, 16, así que no puede darse por cumplida la especificación LTL de la ecuación 17.

11.3. Caso 10.3

La trayectoria resultante obtenida en la figura 15 cumple con la especificación dada a través de la fórmula LTL 18 ya que solo ingresa a las regiones 2 y 15 evadiendo siempre las otras regiones de interés para el sistema. Además, al comparar los resultados depositados en las tablas 4 y 5, se identifica una notable diferencia en el tiempo asociado a la generación del diagrama de estados y transiciones del robot, lo cual repercute directamente en el tiempo asociado al cálculo

de las posibles rutas aceptables que cumplen con la especificación. Esto indica que cuanto más robusta sea la fórmula LTL en términos de la delimitación del comportamiento del espacio de trabajo, menor será el tiempo necesario para el cálculo de una trayectoria solución.

La posibilidad de incluir fórmulas LTL más robustas implica la posibilidad de ejecutar diversos tipos de tareas sobre el mismo entorno, es decir, sujetos a la misma complejidad en la generación del sistema de transiciones del robot. Cuando se habla de diversos tipos de tareas, directamente se hace referencia a las posibilidades de expresión de tareas lógicas que brinda la LTL mostradas en la sección 7.1, principalmente: seguridad, cobertura o secuenciación. La trayectoria calculada en la figura 15 hace referencia a una especificación co-segura y corresponde a la trayectoria con el menor número de transiciones en el autómata del sistema completo. Sin embargo, si se incluyen más regiones de interés, este seguirá siendo el único criterio, es decir, el sistema entregaría la ruta más corta que cubre las 3 regiones sin importar el orden en el que las cubre. Esto da cavida a plantear una primera variación de este caso abordada en la sección 10.3.1 en donde la figura 16 y la tabla 6 expresan la trayectoria y los datos resultantes para la fórmula LTL 19 en la que se incluye la región 12 como una región de interés. Sin embargo, el orden en que se alcanzan las regiones no está especificado y la nueva capacidad de expresividad del algoritmo 2 permite hacerlo cambiando la estructura de cobertura de la fórmula LTL, a una estructura de secuenciación como se muestra en la ecuación 20. Este último cambio permite una nueva variación del caso abordada en la sección 10.3.2 para la cual la fórmula LTL 20 expresa, además del requerimiento de alcanzar en algún momento futuro las regiones 2, 15 y 12, la necesidad de alcanzarlas en ese orden respectivo.

En la trayectoria de la figura 17 se puede notar que las regiones, además de ser alcanzadas en algún momento del tiempo, también son alcanzadas en un orden específico, con una variación en tamaño del autómata Büchi resultante para solucionar la fórmula LTL. Esto comprueba que el algoritmo 2 está sujeto a la capacidad de expresividad de los diferentes tipos de sub-fórmulas LTL que apliquen al caso.

Para el bloque de pruebas que se aborda en la sección 10.3.3, planteado para el algoritmo 2,la especificación dada en lenguaje natural corresponde a “diríjase de la región inicial a la región

final evadiendo los obstáculos" (Lasregiones inicial y final están representadas por estrellas). En los res primeros escenarios de la figura 18 se logra alcanzar el objetivo ya sea evadiendo los obstáculos presentes en la ruta o re calculando una nueva ruta sin obstáculos. Sin embargo, en el escenario ubicado en la parte inferior derecha, el sistema no logra sintetizar un autómata que contenga una ruta válida que sea solución para el problema, es decir, en esta ocasión, el entorno "hizo un movimiento por fuera de las reglas" y por ende no es posible entregar una solución.

11.4. Caso 10.4

El algoritmo 4 que sintetiza políticas de control para equipos de robots idénticos con una única tarea, también está en capacidad de sintetizar una solución para los experimentos anteriores, si se supone que el equipo de robots está conformado por un único robot que debe visitar todas las regiones. Sin embargo, las posibilidades de expresividad en la fórmula LTL utilizada para describir la tarea están limitadas al conjunto de sub-fórmulas co-seguras descritas en 7.2. Esto implica, que en este caso, las tareas de secuenciación no son alcanzables para sistemas con múltiples agentes.

La red de Petri resultante para describir el espacio de trabajo tiene tantos lugares como particiones tenga el espacio. En el caso del entorno mostrado en la figura 11, La red de Petri calculada con el algoritmo 4 y descrita en la tabla 8 tiene 42 lugares, asociados cada uno con las 42 particiones presentes en el entorno y las transiciones están ligadas a la condición de adyacencia entre las regiones en las que se particiona el entorno. Debido a que el espacio de trabajo no va a cambiar si se agregan más robots que ayuden a cumplir la tarea, el tiempo asociado a la construcción de la red de Petri se va a mantener constante sin importar cuantos robos se agreguen al sistema. Para la solución del problema MILP se utiliza el software CPLEX de IBM y la solución al problema está dada por la secuencia de activaciones que se deben presentar en la red de Petri para garantizar que el robot se va a desplazar desde el lugar de inicio hasta el lugar final pasando y evitando las regiones especificadas en la fórmula. Si bien es cierto que en este caso no se pueden especificar tareas de secuenciación, el sistema no se

encuentra reducido únicamente a tareas de cobertura, pues es posible, en algunas ocasiones, especificar el lugar o conjunto de lugares finales que se desea ocupen, o no, los robots.

11.5. Caso 10.5 y caso 10.6

En los casos anteriores se han probado los algoritmos 1, 2 y 4 con un solo robot y se ha demostrado que cumplen con su propósito para el caso de la planificación de trayectorias y la evasión de obstáculos en tiempos aceptables para el contexto.

Recurriendo al objetivo general de esta tesis 3.1, que habla de la posibilidad de utilizar equipos de robots para alcanzar de manera conjunta una tarea específica, se plantea el entorno de la figura 20 con múltiples robots y múltiples regiones de interés. Se genera una especificación LTL que debe ser alcanzada de manera cooperativa por los 2 robots, la cual consiste en visitar en el menor número de transiciones o activaciones las 3 regiones de interés demarcadas en el espacio de trabajo. La fórmula LTL está dada en la ecuación 22 y su forma FNC por la ecuación 23.

El sistema de transiciones para un solo robot está estrechamente ligado a la cantidad de particiones que se generen para describir el espacio de trabajo, que en el caso del entorno 20 es 26, de aquí, que en la tabla 9 ese sea el número de estados del autómata. Sin embargo, debido al principio de multiplicación, el autómata que describe todos los posibles estados del equipo de robots contiene 676 estados (26 de un robot multiplicado por 26 del otro robot). Además, es necesario incluir también la cantidad de estados que contiene el autómata de Büchi que representa la especificación dada como una fórmula LTL. Para este caso, el autómata de Buchi cuenta con 8 estados, lo que generaría como resultado un autómata general para el sistema con 5408 estados y sobre este autómata se hace la evaluación de todas las posibles rutas para obtener una con la menor cantidad transiciones en la que se cumpla la especificación (pueden existir varias rutas que sean solución). Debido a la necesidad de sincronización entre los autómatas de los robots, el autómata general crece y hace más extenso el tiempo de cálculo de la ruta solución para el problema, que en este caso es 11.232 segundos.

Al comparar los datos depositados en las tablas 9 y 10 se identifica claramente que el algoritmo

4 es el que utiliza menos tiempo para obtener las trayectorias que garantizan el cumplimiento de la fórmula LTL. Sin embargo, aún es posible encontrar una solución con los 2 algoritmos para un entorno con 2 robots.

11.6. Caso 10.7

Al intentar encontrar una solución a la fórmula LTL 22 en el entorno de la figura 23 con tres robots, a través del algoritmo 3, se presenta una explosión combinacional de estados que impide que el computador en el que se ejecuta el experimento pueda entregar un resultado que garantice el cumplimiento de la especificación, por este motivo no se entregan resultados de trayectoria ni tabla de datos para esta simulación.

11.7. Caso 10.8

Al comparar los datos depositados en las tablas 10 y 11, se evidencia que la complejidad en la resolución del problema se mantiene estable y no aumenta en consecuencia con la cantidad de robots agregados. Esto se debe a que la estructura de la red de Petri que describe el entorno no cambia con respecto a la cantidad de robots que se utilizan, pues estos representan marcas en los lugares ya existentes de la red.

Para demostrar que el tiempo para encontrar la solución no depende de la cantidad de robots que se agregan al sistema, se plantea un entorno con 6 regiones de interés y se analizan los resultados obtenidos a través del algoritmo 4 con la especificación mostrada en la ecuación 24 para 2, 3, 4 y 5 robots alcanzando de manera cooperativa la especificación.

En la figura 26 se evidencia que en todos los casos se cumple con la especificación. Sin embargo, también se nota que no en todos los casos se desplazan todos los robots, es decir, en algunos casos, unos pocos robots se mueven para alcanzar la especificación mientras los otros permanecen estáticos. Esto se debe a que el algoritmo es óptimo desde el punto de vista de la cantidad de transiciones que se deben activar para alcanzar la especificación, lo que implica

en este caso, que mover todos los robots genera más disparos en las transiciones de los que son necesarios.

Se puede notar claramente en la tabla 12 que los tiempos no cambian con relación a la cantidad de robots ya que al mantenerse la topología de la RdP se mantiene también el tamaño del problema MILP.

11.8. Caso 10.9

Es importante resaltar que las variaciones en el tiempo de cálculo de la solución para el algoritmo 4 se deben a la complejidad de la especificación de la tarea. Sin embargo, debido a que en este trabajo se consideran solo fórmulas LTL co-seguras, la complejidad en la especificación de las tareas es siempre la misma, así que cambiar el orden, las regiones a visitar o las regiones a evadir, no debe modificar considerablemente el tiempo de cálculo de las trayectorias solución.

En la figura 27 se puede notar claramente que los robots cumplen con la especificación dada en este caso, sin pasar por las regiones rosa y gris(E, F) y que además, después de visitar las regiones de interés, salen de la misma para terminar en una región diferente. Sin embargo, al analizar los datos obtenidos en la tabla 13 se puede notar que aunque la especificación global cambió, los tiempos de ejecución se sostienen cercanos a los anteriores y el cálculo de las trayectorias solución no varía considerablemente con respecto a los tiempos para el mismo entorno con 3 robots depositados en la tabla 12.

11.9. Ejemplo de aplicación 10.10

Para el ejemplo de aplicación planteado, se puede notar en la figura 29 que para dar solución al problema, no es necesario utilizar los 8 robots, es decir, utilizando solo 4 camiones recolectores se pueden cubrir todas las áreas en el menor tiempo posible. Además, Los tiempos depositados en la tabla 14 evidencian un incremento sustancial con respecto a los ejercicios anteriores. Esto es debido a que en este caso el entorno creció considerablemente, generando una RdP con 160 lugares y 476 transiciones, lo cual hace que el tamaño del problema MILP aumente. Sin

embargo, en el contexto de la aplicación del algoritmo para resolver el problema, este tiempo es aceptable, pues las rutas se calculan previamente en la planificación de trabajo de los recolectores, es decir, el sistema no necesita reaccionar rápidamente ante cambios inesperados.

Parte V

CONCLUSIONES Y RECOMENDACIONES

12. CONCLUSIONES

La metodología propuesta permite flexibilidad en cuanto al cambio de la tarea global dada a un equipo de robots, pues en el método tradicional que utiliza autómatas en lugar de redes de Petri, es necesario volver a sintetizar un autómata completamente diferente y en este trabajo se demostró que solo es necesario cambiar la tarea dada como una fórmula LTL o FNC. La metodología propuesta resuelve el problema de síntesis de autómatas para el control alto nivel de los vehículos en dónde la planificación de movimientos o el cambio de tareas durante la ejecución es una constante.

Se describe un método que permite sintetizar un autómata para generar un plan de movimiento que logra objetivos especificados por el usuario, expresados en fórmulas de lógica temporal, mostrando que bajo estos criterios se puede mitigar el problema de explosión combinacional de estados. Se mostró que muchos comportamientos complejos de sistemas robóticos pueden ser expresados en lógica temporal y por lo tanto se puede calcular su solución utilizando este método.

Se describe un método que permite sintetizar un autómata para cumplir un comportamiento especificado por el usuario expresado en lógica temporal con la particularidad de incluir comportamientos reactivos en los cuales el robot depende no solo de las especificaciones de la tarea global dada por el usuario, sino también de la información que captura del entorno a través de sus sensores durante el tiempo de ejecución, mostrando que bajo estos criterios se puede

mitigar el problema de explosión combinacional de estados.

El método presentado planifica automáticamente las tareas a ejecutar de un equipo de robots móviles que cooperan para alcanzar una tarea dada como una fórmula LTL co-segura o una fórmula Booleana en forma FNC que se genera con base en las regiones de interés demarcadas en el entorno de trabajo de los robots. La especificación se proporciona sobre regiones de interés del entorno, y expresa el comportamiento deseado de todo el equipo, sin asignaciones individuales de robots a partes de la tarea.

El adecuado funcionamiento de este método se basa en la capacidad de generar una abstracción del equipo y el entorno en forma de una RMPN. A diferencia de los enfoques que usan diagramas de estados y transiciones y productos sincrónicos entre estos, el modelo RMPN tiene una topología fija frente al tamaño del equipo, es decir, la cantidad de robots que se agreguen al sistema no afectan la topología de la RdP.

Cuando se logra representar la fórmula booleana en forma FNC y se representa esta a través de un conjunto de desigualdades lineales, las evaluaciones de estas variables se vinculan con una secuencia finita de marcas de la RdP y se obtiene una formulación de un problema MILP, cuya solución proporciona una secuencia de activaciones de las transiciones de la RdP que se convierten en trayectorias robóticas individuales. Esto garantiza, que la solución encontrada es óptima con respecto al número de transiciones seguidas por los miembros del equipo.

Se compararon dos algoritmos de solución para entornos con múltiples regiones de interés y múltiples robots y se identificaron los problemas computacionales asociados a la necesidad de sincronización de los sistemas de transiciones cuando estos se utilizan para modelar el entorno. También se evidenció en los experimentos, los beneficios obtenidos con la metodología propuesta ya que los robots pueden seguir sus trayectorias sin necesidad de sincronizarse con otros miembros del equipo.

13. RECOMENDACIONES Y TRABAJO FUTURO

Capturar las especificaciones dadas para los sistemas y para el entorno en fórmulas LTL es un proceso complejo, por lo cual, es bueno especificar una metodología que facilite la generación de estas fórmulas minimizando los errores que se pueden cometer por parte del usuario.

Otra dirección en la que se está trabajando es en la evaluación de esta metodología para resolver problemas en sistemas reactivos a cambios no planeados del entorno, lo cual tiene gran aplicabilidad en sistemas robóticos con 2 y 3 grados de libertad como puede ser la planificación de movimientos en drones.

Las extensiones futuras de este método pueden incluir un método para reducir la carga computacional que se puede reducir mediante el uso de una adaptación subóptima que resuelve el problema en un sistema RdP reducido.

Se trabajará en el futuro para generalizar las especificaciones a un lenguaje LTL más expresivo y para evaluar la viabilidad computacional para escenarios complejos. Además, se desarrollarán resultados experimentales más elaborados para ilustrar las ventajas de la solución propuesta.

14. PRODUCTOS

A continuación se listan algunos productos e investigación cuyo desarrollo permitió la consolidación de la metodología presentada en este trabajo.

- Conferencia de apertura del Segundo Congreso Internacional de Electromecánica y Eléctrica en Latacunga - Ecuador. (Algoritmo 1).
- Artículo de investigación publicado en las memorias del Segundo Congreso Internacional de Electromecánica y Eléctrica en Latacunga - Ecuador. (Algoritmo 1) [65].
- Ponencia en el 3er Colombian Conference on Automatic Control (CCAC 2017) en Cartagena - Colombia. (Algoritmo 2).

- Artículo de investigación publicado en IEEE Xplore como memoria del 3er Colombian Conference on Automatic Control (CCAC 2017) en Cartagena - Colombia. (Algoritmo 2)[67].
- Un software portable multiplataforma (PINET) presentao ante la Universidad Tecnológica de Pereira.
- Artículo de investigación en proceso de evaluación para publicación en IEEE Xplore - Colombia. (Algoritmo 4).
- Artículo de investigación en proceso de evaluación para publicación en la revista Scientia et Technica. (Fase 2).
- Tesis de Maestría para la Maestría en Ingeniería Eléctrica de la Universidad Tecnológica de Pereira. (Algoritmo 5).

BIBLIOGRAFÍA

[1] KRESS-GAZIT, Hadas. Robot challenges: Toward development of verification and synthesis techniques [from the guest editors]. En: IEEE Robotics & Automation Magazine, tomo 18, nº 3, 2011, págs. 22–23. 1

[2] GNING, Amadou, *et al.* Evolving networks for group object motion estimation. En: Target Tracking and Data Fusion: Algorithms and Applications, 2008 IET Seminar on. IET, 2008, págs. 99–106. 1

[3] REALPE, M.; VINTIMILLA, B. X. y VLACIC, L. A Fault Tolerant Perception system for autonomous vehicles. En: 2016 35th Chinese Control Conference (CCC), 2016, págs. 6531–6536. 1

[4] WONGPIROMSARN, Tichakorn; TOPCU, Ufuk y MURRAY, Richard M. Receding horizon temporal logic planning. En: IEEE Transactions on Automatic Control, tomo 57, nº 11, 2012, págs. 2817–2830. 1

[5] CHOSET, Howie M. Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005. 1

[6] LAVALLE, Steven M. Planning algorithms. Cambridge university press, 2006. 1, 8.3

[7] DING, Xuchu, *et al.* Optimal control of Markov decision processes with linear temporal logic constraints. En: IEEE Transactions on Automatic Control, tomo 59, nº 5, 2014, págs. 1244–1257. 1, 2

[8] BELTA, Calin, *et al.* Symbolic planning and control of robot motion [grand challenges of robotics]. En: IEEE Robotics & Automation Magazine, tomo 14, nº 1, 2007, págs. 61–70. 1

[9] FRANCESCHELLI, Mauro, *et al.* Gossip algorithms for heterogeneous multi-vehicle routing problems. En: Nonlinear Analysis: Hybrid Systems, tomo 10, 2013, págs. 156–174. 1

[10] DING, Xu Chu, *et al.* Automatic deployment of robotic teams. En: IEEE Robotics & Automation Magazine, tomo 18, nº 3, 2011, págs. 75–86. 1

[11] LATOMBE, Jean-Claude. Robot motion planning, tomo 124. Springer Science & Business Media, 2012. 1

[12] VIDE, Carlos Martín. Gramáticas formales (y similares) para lingüistas. En: Lenguajes naturales y lenguajes formales: actas del X congreso de lenguajes naturales y lenguajes formales:(Sevilla, 26-30 de septiembre de 1994). Promociones y Publicaciones Universitarias, PPU, 1994, págs. 71–92. 1, 5.3

[13] CHOMSKY, Noam. Aspects of a theory of syntax. En: Cambridge, Mass: MI'I' I'rw Language Journal, tomo 53, 1965, págs. 334–341. 1, 5.3

[14] PARTEE, Barbara BH; TER MEULEN, Alice G y WALL, Robert. Mathematical methods in linguistics, tomo 30. Springer Science & Business Media, 2012. 1

[15] HOPCROFT, John E; MOTWANI, Rajeev y ULLMAN, Jeffrey D. Teoría de autómatas, lenguajes y computación. En: Pearson educación, 2007. 1, 5.1, 5.2

[16] BRENA, Ramon F. Autómatas y lenguajes. Tecnológico de Monterrey, 2003. 1, 5.1, 5.2, 5.5

[17] TORRES, Sergio Augusto Cardona. Lógica matemática para ingeniería de sistemas y computación. ELIZCOM SAS, 2010. 1

[18] DING, X., *et al.* Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. En: IEEE Transactions on Automatic Control, tomo 59, nº 5, 2014, págs. 1244–1257. ISSN 0018-9286. 1

[19] VALMARI, Antti. The state explosion problem. En: Lectures on Petri nets I: Basic models. Springer, 1998, págs. 429–528. 1

[20] BELTA, Calin, *et al.* Symbolic planning and control of robot motion [grand challenges of robotics]. En: IEEE Robotics & Automation Magazine, tomo 14, nº 1, 2007, págs. 61–70. 1

[21] BHATIA, Amit; KAVRAKI, Lydia E y VARDI, Moshe Y. Sampling-based motion planning with temporal goals. En: Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010, págs. 2689–2696. 1

[22] WONGPIROMSARN, Tichakorn; TOPCU, Ufuk y MURRAY, Richard M. Receding horizon temporal logic planning for dynamical systems. En: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on. IEEE, 2009, págs. 5997–6004. 2

[23] KARAMAN, Sertac y FRAZZOLI, Emilio. Sampling-based motion planning with deterministic μ-calculus specifications. En: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on. IEEE, 2009, págs. 2222–2229. 2

[24] CLARKE, Edmund M; GRUMBERG, Orna y PELED, Doron. Model checking. MIT press, 1999. 2

[25] CIMATTI, Alessandro, *et al.* Nusmv 2: An opensource tool for symbolic model checking. En: International Conference on Computer Aided Verification. Springer, 2002, págs. 359–364. 2

[26] GERARD, J Holzmann. The SPIN model checker: Primer and reference manual, 2003. 2

[27] GUO, Meng y DIMAROGONAS, Dimos V. Multi-agent plan reconfiguration under local LTL specifications. En: The International Journal of Robotics Research, tomo 34, nº 2, 2015, págs. 218–235. 2, 4

[28] KLOETZER, Marius y MAHULEA, Cristian. Ltl-based planning in environments with probabilistic observations. En: IEEE Transactions on Automation Science and Engineering, tomo 12, nº 4, 2015, págs. 1407–1420. 2, 4

[29] KLOETZER, Marius y MAHULEA, Cristian. Multi-robot path planning for syntactically co-safe LTL specifications. En: Discrete Event Systems (WODES), 2016 13th International Workshop on. IEEE, 2016, págs. 452–458. 2, 4

[30] KRESS-GAZIT, Hadas; FAINEKOS, Georgios E y PAPPAS, George J. Temporal-logic-based reactive mission and motion planning. En: IEEE transactions on robotics, tomo 25, nº 6, 2009, págs. 1370–1381. 4, 7.2

[31] ULUSOY, Alphan, *et al.* Robust multi-robot optimal path planning with temporal logic constraints. En: Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012, págs. 4693–4698. 4

[32] COSTELHA, Hugo y LIMA, Pedro. Robot task plan representation by Petri nets: modelling, identification, analysis and execution. En: Autonomous Robots, tomo 33, nº 4, 2012, págs. 337–360. 4

[33] KLOETZER, Marius y MAHULEA, Cristian. A Petri net based approach for multi-robot path planning. En: Discrete Event Dynamic Systems, tomo 24, nº 4, 2014, págs. 417–445. 4

[34] MAHULEA, Cristian y KLOETZER, Marius. Planning mobile robots with Boolean-based specifications. En: Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on. IEEE, 2014, págs. 5137–5142. 4, 9.3

[35] ROSZKOWSKA, Elzbieta y REVELIOTIS, Spyros. A distributed protocol for motion coordination in free-range vehicular systems. En: Automatica, tomo 49, nº 6, 2013, págs. 1639–1653. 4

[36] KLOETZER, Marius; MAHULEA, Cristian y COLOM, José-Manuel. Petri net approach for deadlock prevention in robot planning. En: Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on. IEEE, 2013, págs. 1–4. 4

[37] LI, ZhiWu; WU, NaiQi y ZHOU, MengChu. Deadlock control of automated manufacturing systems based on Petri netsâA literature review. En: IEEE Transactions on Systems, Man,

and Cybernetics, Part C (Applications and Reviews), tomo 42, nº 4, 2012, págs. 437–462. 4

[38] GUO, Meng y DIMAROGONAS, Dimos V. Bottom-up motion and task coordination for loosely-coupled multi-agent systems with dependent local tasks. En: Automation Science and Engineering (CASE), 2015 IEEE International Conference on. IEEE, 2015, págs. 348–355. 4

[39] KARAMAN, Sertac y FRAZZOLI, Emilio. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. En: International Journal of Robust and Nonlinear Control, tomo 21, nº 12, 2011, págs. 1372–1395. 4

[40] CHEN, Yushan, *et al.* Formal approach to the deployment of distributed robotic teams. En: IEEE Transactions on Robotics, tomo 28, nº 1, 2012, págs. 158–171. 4

[41] TUMOVA, Jana y DIMAROGONAS, Dimos V. Decomposition of multi-agent planning under distributed motion and task LTL specifications. En: Decision and Control (CDC), 2015 IEEE 54th Annual Conference on. IEEE, 2015, págs. 7448–7453. 4

[42] SCHILLINGER, Philipp; BÜRGER, Mathias y DIMAROGONAS, Dimos. Decomposition of Finite LTL Specifications for Efficient Multi-Agent Planning. En: 13th International Symposium on Distributed Autonomous Robotic Systems, London, November 6-9, 2016., 2016. 4

[43] ARAGUES, Rosario, *et al.* Distributed algebraic connectivity estimation for adaptive event-triggered consensus. En: American Control Conference (ACC), 2012. IEEE, 2012, págs. 32–37. 4

[44] FRANCESCHELLI, Mauro; GIUA, Alessandro y PISANO, Alessandro. Finite-time consensus on the median value by discontinuous control. En: American Control Conference (ACC), 2014. IEEE, 2014, págs. 946–951. 4

[45] GARRIDO, Santiago, *et al.* General path planning methodology for leader-follower robot formations. En: International Journal of Advanced Robotic Systems, tomo 10, nº 1, 2013, pág. 64. 4

[46] GUTIERREZ, Alvaro Angel Orozco; LONDONO, German Andres Holguin y LONDOÑO, Mauricio Holguin. Fundamentos teoricos para los automatas industriales. Universidad Tecnologica de Pereira, 2010. 5.3, 5.5

[47] MORENO, Manuel Alfonseca; RODRíGUEZ, Justo Sancho y ORGA, Miguel Martínez. Teoría de lenguajes, gramáticas y autómatas, 1987. 5.3

[48] MURATA, T. Petri nets: Properties, analysis and applications. En: Proceedings of the IEEE, tomo 77, nº 4, 1989, págs. 541–580. ISSN 0018-9219. 5.4

[49] SILVA, Manuel; TERUE, Enrique y COLOM, José Manuel. Linear algebraic and linear programming techniques for the analysis of place/transition net systems. En: Advanced Course on Petri Nets. Springer, 1996, págs. 309–373. 5.4

[50] REVIEWER-GASARCH, William. BOOK REVIEW: Finite Automata, Formal Logic, and Circuit Complexity. By Howard Straubing.(Birkhauser. 1994. xii+ 226pp. ISBN 0-8176-3719-2.). En: ACM SIGACT News, tomo 25, nº 3, 1994, págs. 28–32. 5.5

[51] WONGPIROMSARN, Tichakorn; TOPCU, Ufuk y MURRAY, Richard M. Receding horizon control for temporal logic specifications. En: Proceedings of the 13th ACM international conference on Hybrid systems: computation and control. ACM, 2010, págs. 101–110. 5.5

[52] WONGPIROMSARN, Tichakorn y MURRAY, Richard M. Distributed mission and contingency management for the DARPA urban challenge. En: International Workshop on Intelligent Vehicle Control Systems (IVCS), 2008. 5.5

[53] EMERSON, E Allen. Temporal and modal logic. En: Formal Models and Semantics. Elsevier, 1990, págs. 995–1072. 7.2, 7.2

[54] MANNA, Zohar y PNUELI, Amir. The temporal logic of reactive and concurrent systems: Specification. Springer Science & Business Media, 2012. 7.2

[55] PITERMAN, Nir; PNUELI, Amir y SAÂAR, Yaniv. Synthesis of reactive (1) designs. En: International Workshop on Verification, Model Checking, and Abstract Interpretation. Springer, 2006, págs. 364–380. 7.2, 7.2, 9.2

[56] KUPFERMAN, Orna y VARDI, Moshe Y. Model checking of safety properties. En: Formal Methods in System Design, tomo 19, nº 3, 2001, págs. 291–314. 7.2

[57] VALENCIA, J. L. M.; LONDONO, M. H. y MEJIA, A. E. A methodology to evaluate combinatorial explosion using LTL in autonomous ground navigation applications. En: 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC), 2017, págs. 1–6. 8.1

[58] GRUMBERG, Orna; CLARKE, EM y PELED, Doron. Model checking, 1999. 8.3

[59] WOLPER, Pierre; VARDI, Moshe Y y SISTLA, A Prasad. Reasoning about infinite computation paths. En: Foundations of Computer Science, 1983., 24th Annual Symposium on. IEEE, 1983, págs. 185–194. 8.3

[60] HOLZMANN, Gerard. Spin model checker, the: primer and reference manual. Addison-Wesley Professional, 2003. 8.3

[61] GASTIN, Paul y ODDOUX, Denis. Fast LTL to Büchi automata translation. En: International Conference on Computer Aided Verification. Springer, 2001, págs. 53–65. 8.3

[62] BROWN, Frank Markham. Boolean reasoning: the logic of Boolean equations. Springer Science & Business Media, 2012. 8.3

[63] SMAUS, Jan-Georg. On Boolean functions encodable as a single linear pseudo-Boolean constraint. En: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. Springer, 2007, págs. 288–302. 8.3

[64] ZHENG, Yongyan; ZHOU, Jiong y KRAUSE, Paul. A model checking based test case generation framework forweb services. En: Information Technology, 2007. ITNG'07. Fourth International Conference on. IEEE, 2007, págs. 715–722. 9.1

[65] JORGE MARTÍNEZ, Mauricio Holguín y ESCOBAR, Andrés. Planificación de movimientos para sistemas de navegación autónoma terrestre utilizando lógica temporal lineal. En: Memorias Científicas del II Congreso Internacional Electromecánica y Eléctrica, 2017. 9.1, 14

[66] PNUELI, Amir y ROSNER, Roni. On the synthesis of a reactive module. En: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. ACM, 1989, págs. 179–190. 9.2

[67] VALENCIA, Jorge Luis Martinez; LONDOÑO, Mauricio Holguín y MEJíA, Andrés Escobar. A methodology to evaluate combinatorial explosion using LTL in autonomous ground navigation applications. En: Automatic Control (CCAC), 2017 IEEE 3rd Colombian Conference on. IEEE, 2017, págs. 1–6. 9.2, 14

[68] YEN, Jin Y. Finding the k shortest loopless paths in a network. En: management Science, tomo 17, nº 11, 1971, págs. 712–716. 9.3

[69] GONZÁLEZ, Ramón; MAHULEA, Cristian y KLOETZER, Marius. A Matlab-Based Interactive Simulator for Mobile Robotics. En: IEEE CASE'2015: Int. Conf. on Autom. Science and Engineering, 2015. IV